Robuste Zustandsschätzung zur Navigation und Regelung autonomer und bemannter Multikopter mit verteilten Sensoren

Heft 54
Schriftenreihe der Fachrichtung Geodäsie
Fachbereich Bau- und Umweltingenieurwissenschaften
Technische Universität Darmstadt
ISBN 978-3-935631-43-3

Darmstadt, April 2020

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Heft 54

Darmstadt, April 2020

Jan Zwiener

Robuste Zustandsschätzung zur Navigation und Regelung autonomer und bemannter Multikopter mit verteilten Sensoren

Schriftenreihe
Fachrichtung Geodäsie
Fachbereich Bau- und Umweltingenieurwissenschaften
Technische Universität Darmstadt

ISBN 978-3-935631-43-3

Schriftenreihe Fachrichtung Geodäsie der Technischen Universität Darmstadt

Online unter: https://tuprints.ulb.tu-darmstadt.de/id/eprint/11668

Diese Arbeit ist gleichzeitig veröffentlicht in der Reihe C der
Deutschen Geodätischen Kommission, München 2019.

2. Auflage: Es handelt sich um eine leicht überarbeitete Fassung zur Vereinheitlichung der Nomenklatur der verschiedenen Zustandsschätzer

Verantwortlich für die Herausgabe der Schriftenreihe:

Der Sprecher der Fachrichtung Geodäsie
im Fachbereich Bau- und Umweltingenieurwissenschaften
der Technischen Universität Darmstadt

Bezugsnachweis:

Technische Universität Darmstadt
Institut für Geodäsie
Franziska-Braun-Str. 7
64287 Darmstadt

ISBN: 978-3-935631-43-3

Robuste Zustandsschätzung zur Navigation und Regelung autonomer und bemannter Multikopter mit verteilten Sensoren

Vom Fachbereich Bau- und Umweltingenieurwissenschaften
der Technischen Universität Darmstadt
zur Erlangung des akademischen Grades eines
Doktor-Ingenieurs (Dr.-Ing.) genehmigte Dissertation

vorgelegt von
Jan Zwiener, M. Sc.
aus Sindelfingen

Referent: Prof. Dr.-Ing. Reiner Jäger
Korreferent: Prof. Dr.-Ing. Matthias Becker
Korreferent: Prof. Dr.-Ing. Uwe Klingauf
Tag der Einreichung: 20. November 2018
Tag der mündlichen Prüfung: 8. Februar 2019

Darmstadt, April 2020
D17

Zusammenfassung

Die robuste Navigation von unbemannten aber auch bemannten VTOL Multirotor-Fluggeräten mit vier oder mehr Motoren und Festpropellern steht im Fokus dieser Arbeit. Dabei wird auf leichtgewichtige, platzsparende, kostengünstige und stromsparende Sensorik aufgebaut. Eine neue Ausprägung des Kalman-Filters wird vorgestellt: das Simplex Kalman-Filter mit robusten $\mathscr{L}_1$ Eigenschaften.

Dieser Ansatz erlaubt darüber hinaus die Navigationszustandsschätzung unter der Hinzunahme von Ungleichungen und Bedingungsgleichungen. Somit kann, bei vorliegenden Informationen über das Gesamtsystem oder der Umgebung, der Lösungsraum auf einfache Art eingeschränkt werden. Es wird gezeigt, wie das Simplex Kalman-Filter mit Hilfe des binären Raumteilungsverfahrens (Binary Space Partitioning) die Einbeziehung von beliebigen konkaven Raumgeometrien zulässt. Somit stellt das Verfahren beispielsweise für manche Anwendungsfälle eine Alternative zu einem Partikelfilter dar. Die Auswirkungen von Ungleichungen auf die Wahrscheinlichkeitsdichtefunktion wird betrachtet.

Zudem wird auf Basis der Simplex $\mathscr{L}_1$ Methode eine redundante Flugregelungsarchitektur für unbemannte und bemannte Multirotor-Fluggeräte vorgestellt. Diese Architektur erlaubt die Einbindung beliebig vieler dissimilarer Flugsteuerungsrechner und dezentralisiert die Entscheidung, welcher dieser Rechner aktiv die Regelung des Fluggeräts übernimmt.

Abstract

The robust navigation of unmanned as well as manned VTOL multi rotor aircraft with four or more engines and fixed pitch propellers is the focus of this work. It is based on lightweight, space-saving, cost-effective and energy-saving sensor technology. A new extension of the Kalman filter is introduced: the Simplex Kalman filter with robust $\mathscr{L}_1$ properties.

This approach also allows the estimation of the navigational state vector under the addition of linear inequality constraints and equality constraints. Thus, with information available about the overall system or the environment, the solution space can be easily restricted. The Simplex Kalman filter can use the so called binary space partitioning method (BSP) to restrict the solution space to arbitrary geometries. For some use cases this is an alternative to a particle filter. The effects of inequalities on the probability density function is considered.

In addition, based on the Simplex $\mathscr{L}_1$ method, a redundant flight control architecture for unmanned and manned multirotor aircraft is presented. This architecture allows the integration of any number of dissimilar flight control computers and decentralizes the decision, which of these computers actively controls the aircraft in a closed loop behaviour.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

$\mathscr{L}_1$, $\mathscr{L}_2$	L_1-Norm bzw. L_2-Norm
$\mathscr{L}_\infty$	L_∞-Norm bzw. Tschebycheff-Norm
AD	Allan Deviation
AHRS	Attitude and Heading Reference System
ARW	Angular Random Walk
AVAR	Allan Varianz
BRW	Bias Random Walk
BS	Bias Stability
CKF	Cubature Kalman-Filter
DGPS	Differential GPS
ECEF	Earth Centered, Earth Fixed
ECI	Earth Centered Inertial
EGM 2008	Earth Gravitational Model 2008
EKF	Extended Kalman-Filter
ESC	Electronic Speed Controller
FOG	Fibre Optic Gyroscope
GHM	Gauß-Helmert-Modell
GMM	Gauß-Markov-Modell
GNSS	Global Navigation Satellite Systems
GPS	Global Positioning System
HMM	Hidden Markov Model
IIR	Infinite Impulse Response
ILS	Integer Least Squares
IMU	Inertial Measurement Unit
INS	Inertial Navigation System
IP	Integer Programming
ITRF	International Terrestrial Reference Frame
L1, L2, L5	GNSS Trägerfrequenzen
LP	Linear Programming
MEMS	Microelectromechanical System
MOEMS	Micro Opto Electro Mechanical Systems
MRAC	Model Reference Adaptive Control
NED	North East Down System (Navigationsframe)
PDF	Probability Density Function
PID	Proportional-Integral-Derivative
PPP	Precise Point Positioning

PSR	Pseudorange
PWM	Pulsweitenmodulation
RLG	Ring Laser Gyroscope
SBAS	Space-based Augmentation System
SNR	Signal-To-Noise Ratio
SPKF	Sigma-Point Kalman-Filter
TEC	Total Electron Content
TOW	Takeoff Weight
UAS	Unmanned Aerial System
UAV	Unmanned Aerial Vehicle
UKF	Unscented Kalman-Filter
VTOL	Vertical take-off and landing
WGS 84	World Geodetic System 1984
WMM 2015	World Magnetic Model 2015
ZRU	Zero Rotation Update
ZUPT	Zero Velocity Update

Notation

Allgemein	
Skalar	Kursive Kleinbuchstaben, z. B. x, t, ψ
Winkel	Kursive griechische Buchstaben, z. B. α, β, γ
Vektoren	Kleinbuchstaben (fett), z. B. $\mathbf{v}$
Betrag von Vektor	Betragsstriche $\lvert\mathbf{v}\rvert$
Transponierter Vektor	Hochstehendes T, z. B. $\mathbf{v}^T$
Kreuzprodukt	Vektor × Vektor, z. B. $\mathbf{v}_1 \times \mathbf{v}_2$
Kreuzproduktbildende Matrix	$[(\ldots)\times]$: $[\mathbf{v}_1\times]\mathbf{v}_2 = \mathbf{v}_1 \times \mathbf{v}_2$
Zeitliche Ableitung	Punkt Notation: $\frac{\mathrm{d}}{\mathrm{dt}}x(t) = \dot{x}(t)$
Quaternion	Kleinbuchstabe $\mathbf{q} = (q_0, q_1, q_2, q_3)^T$
Realteil von Quaternion	q_0
Imaginärteil von Quaternion	$(q_1, q_2, q_3)^T$
Quaternionenmultiplikation	$\bullet$
Matrix	Großbuchstaben (fett), z. B. $\mathbf{M}$
Transponierte Matrix	Hochstehendes T, z. B. $\mathbf{M}^T$
Einheitsmatrix	$\mathbf{I}$
Einheitsmatrix mit n Zeilen u. m Spalten	$\mathbf{I}_{n\times m}$
Nullmatrix mit n Zeilen u. m Spalten	$\mathbf{0}_{n\times m}$
Diagonalmatrix	$\mathrm{diag}(\ldots)$
Signumfunktion	$\mathrm{sgn}(\ldots)$
Determinante einer Matrix	$\det(\ldots)$
Vier Quadranten Arkustangens	$\arctan2(y, x)$
Einheiten	Aufrechtstehende Buchstaben, z. B. °C, kg, kJ
Infinitesimaler Zeitschritt	dt
Diskreter Zeitschritt	Δt

Navigation	
Komponente auf der X-Achse	$(\ldots)_x$
Komponente auf der Y-Achse	$(\ldots)_y$
Komponente auf der Z-Achse	$(\ldots)_z$
Komponente in Nordrichtung (North)	$(\ldots)_n$
Komponente in Ostrichtung (East)	$(\ldots)_e$
Vertikalkomponente nach unten (down)	$(\ldots)_d$
Inertialkoordinatensystem	$(\ldots)^i$
Navigationskoordinatensystem (n-frame)	$(\ldots)^n$

Erdfestes Koordinatensystem (ECEF) (e-frame)	$(\dots)^e$
Körperfestes Koordinatensystem (b-frame)	$(\dots)^b$
Plattform Koordinatensystem (p-frame)	$(\dots)^p$
Sensor Koordinatensystem	$(\dots)^s$
Leverarm/Hebelarm/Offset für verteilte Sensoren	$\mathbf{o}$
Rollwinkel (Roll)	Griechischer Buchstabe Phi ϕ
Nickwinkel (Pitch)	Griechischer Buchstabe Theta θ
Gierwinkel / Azimut / Heading (Yaw)	Griechischer Buchstabe Psi ψ
Rotationsmatrix	$\mathbf{R}_q^z$ (von System q nach System z)
Drehratenvektor	$\boldsymbol{\omega}_{bc}^a$ (c gegenüber b, abgebildet in a)
Kreuzproduktbildende Matrix aus Drehraten	$\boldsymbol{\Omega}_{bc}^a = [\boldsymbol{\omega}_{bc}^a \times]$
Navigationszustandsvektor	$\mathbf{y}(t)$
Positionsvektor	$\mathbf{x}(t)$
Geschwindigkeitsvektor	$\dot{\mathbf{x}}(t)$
Beschleunigungsvektor	$\ddot{\mathbf{x}}(t)$
Geographische Breite (*latitude*)	B
Geographische Länge (*longitude*)	L
Ellipsoidische Höhe	h
Orthometrische Höhe	H
Barometrische Höhe	h^b
Breite (auf Kugel)	φ
Länge (auf Kugel)	λ
Lichtgeschwindigkeit	c (299 792 458 m/s)
Schwerebeschleunigung (engl. *gravity*)	$\mathbf{g}$
Massenanziehung (engl. *gravitation*)	$\mathbf{g}_{\text{Gravitation}}$
Zentripetalbeschleunigung	$\mathbf{g}_{\text{Zentrifugal}}$
Spezifische Kraft (engl. *specific force*)	$\mathbf{f}$ (allg.: $\mathbf{f}^i = \ddot{\mathbf{x}}^i - \mathbf{g}^i$)

Parameterschätzung

Messvektor	$\mathbf{l}$
Residuum	v
Residuenvektor	$\mathbf{v}$
Wahrer Fehler	Epsilon ϵ
Geschätzte Größe	Zirkumflex $\hat{\mathbf{l}} = \mathbf{l} + \mathbf{v}$
Wahre Größe	Tilde $\tilde{\mathbf{l}} = \mathbf{l} - \epsilon$
Designmatrix	$\mathbf{A}$
Kovarianzmatrix / Dispersionsmatrix	$\mathbf{C}$
Kofaktormatrix	$\mathbf{Q}$
Homogenisierte Matrix	Makron $\bar{\mathbf{A}}$
Erwartungswert	$E[\dots]$
Standardabweichung	σ

Empirische Standardabweichung	s

Flugphysik und Regelungstechnik

System Matrix (zeitkontinuierlich)	$\mathbf{F}$
Prädiktionsmatrix / Transitionsmatrix	$\mathbf{\Phi}$
Steuermatrix	$\mathbf{B}$
Stellgröße	$\mathbf{u}$
Regelabweichung	$\mathbf{e}$
Gewichtsmatrix	$\mathbf{W}$
Trägheitsmoment	$\mathbf{J}$
Drehmoment	$\mathbf{m}$
Schubvektor	$\mathbf{f}$
Motormatrix	$\mathbf{K}$
PID-Regler Faktoren	k_P (proportional), k_I (integral), k_D (derivativ)

1 Einleitung

1.1 Ziele und Beiträge der Arbeit

Seit mehr als zehn Jahren sind kleine VTOL[1] Multirotor-Fluggeräte mit vier oder mehr Motoren und Festpropellern das Thema zahlreicher Forschungsbeiträge verschiedenster Disziplinen. Der Schritt, diese Fluggeräte auf eine bemannte Größe zu skalieren, liegt nahe und erste Konzepte wurden schon vor über 100 Jahren gebaut, allerdings ohne sich durchsetzen zu können. Die Gründe dafür sind vielfältig, aber ein zentraler Punkt ist die Steuerbarkeit. Während sich der mechanische Aufbau der Fluggeräte vereinfacht, erhöht sich der Anspruch an einen menschlichen Piloten derartig, dass ein sicherer Betrieb nicht gewährleistet ist. Rechnergestützte Regelalgorithmen bringen die notwendige Abhilfe, aber Regelalgorithmen sind nur so gut und sicher wie ihre Eingangsdaten. Eine genaue und verlässliche Zustandsstandsschätzung bildet daher die notwendige Grundlage. Mit zunehmender Größe des Fluggeräts rückt besonders der Aspekt der Sicherheit und Robustheit in den Fokus. Aufgrund der i. A. fehlenden Auftriebsflächen sind VTOL Multirotor-Fluggeräte tendenziell eher für kürzere Strecken geeignet, womit dafür aber hingegen einhergeht, dass diese Art von Fluggeräten nicht zu groß gebaut werden. Daher besteht gerade in diesem Segment von vergleichsweise leichten Fluggeräten ein Bedarf an Zustandsschätzungsarchitekturen, die auf leichten und preisgünstigen Sensoren aufbauen, inhärent sicher und robust sind sowie eine angemessene Genauigkeit mit sich bringen.

Die vorliegende Arbeit entstand z. T. im Rahmen eines dreijährigen Forschungsprojekts zwischen dem Institut für Angewandte Forschung (IAF) an der Hochschule Karlsruhe (HSKA) und der e-volo GmbH aus Bruchsal sowie weiteren Teilnehmern. Gefördert wurde das Projekt durch das *Bundesministerium für Wirtschaft und Technologie (BMWi)*[2] mit dem Ziel ein elektrisches VTOL Multirotor-Fluggerät für zwei Personen zu entwickeln.

Die Motivation bzw. der Fokus dieser Arbeit liegt daher auf der Navigationszustandsschätzung für diese neuartige Klasse von Fluggeräten. In diesem Bereich soll eine Lücke zwischen Navigationslösungen für kleine und unbemannte UAVs und Navigationslösungen der traditionellen zivilen Luftfahrt geschlossen werden. Insbesondere durch einen neuen Ansatz zur Zustandsschätzung in Form eines Simplex Kalman-Filters, das Methoden aus der Geodäsie (besonders den M-Schätzern) und dem Bereich Operations Research verbindet. Die zugrunde liegenden Gleichungssysteme werden nicht wie gewöhnlich über die Methoden der linearen Algebra in Matrixform gelöst, sondern über den Simplex Algorithmus, der auf George Dantzig zurückgeht und verbessert wurde durch die Arbeiten von I. Barrodale und F. D. K. Roberts. Der ursprünglich in der Programmiersprache Fortran geschriebene Barrodalle und Roberts Algorithmus wurde im Rahmen der Arbeit in C++ und als MATLAB MEX Modul umgesetzt, um sinnvoll für die Untersuchungen des Simplex Kalman-Filters genutzt werden zu können.

[1] Vertical take-off and landing.
[2] Förderkennzeichen: *VP2957702TL2*. Heute: *Bundesministerium für Wirtschaft und Energie*.

Es ist zu erwarten, dass bemannte Multirotor-Fluggeräte besonders als Flugtaxi in urbanen Gebieten zum Einsatz kommen werden. Für die Untersuchung wurde daher ein Hauptaugenmerk auf GNSS Pseudorangemessungen gelegt, da hier im innerstädtischen Bereich in der Nähe von hohen Gebäuden regelmäßig mit Messfehlern zu rechnen ist. Ein wichtiger Punkt in diesem Zusammenhang ist die Beachtung der Hebelarmgleichungen für die Navigationszustandsschätzung. Bei größeren Fluggeräten ist es nicht immer möglich, dass alle Sensoren im Schwerpunkt installiert sind.

Ein neues Merkmal ist darüber hinaus, dass der vorgestellte Algorithmus um ein System zur binären Raumteilung erweitert wurde (Simplex BSP: Simplex Binary Space Partitioning). Dieses System erlaubt das Einschränken des Lösungsraumes um beliebige konkave 2D oder 3D Räume direkt im Simplex Kalman-Filter selbst. Das können z. B. Innenräume sein, aber auch Straßenkarten. Das Verfahren ist allgemeingültig und kann in vielen Anwendungen von Vorteil sein. Derartiges Wissen, z. B. aus vorhandenem Kartenmaterial, lässt sich sonst meist nur mit aufwendigeren Schätzverfahren einbringen, wie beispielsweise einem rechenintensiven Partikelfilter.

Darüber hinaus wird der Simplex Algorithmus für die Regelung eines bemannten Multirotor Fluggeräts selbst genutzt. In der bemannten Luftfahrt werden nur Architekturen zugelassen, die eine sehr niedrige Gesamtsystemausfallwahrscheinlichkeit nachweisen können ($\leq 10^{-7}$ pro Flugstunde), was oft nur durch verteilte und redundante Sensoren sowie Recheneinheiten zu verwirklichen ist. Durch die $\mathscr{L}_1$ Eigenschaften des Simplex Algorithmus können die Entscheidungen und Ergebnisse von unabhängig zueinander rechnenden verteilten Systemen wieder zusammengeführt werden.

1.2 Gliederung der Arbeit

Kapitel 2 betrachtet die notwendigen Grundlagen der Parameter und Zustandsschätzung für das Simplex Kalman-Filter. Besonders die M-Schätzer Theorie in Abschnitt 2.5 stellt eine wichtige Basis dar. Es wird gezeigt, dass das Kalman-Filter in eine für Kapitel 5 notwendige Form gebracht werden kann.

Kapitel 3 behandelt die verwendeten Methoden und Sensoren der Navigation; darauf baut dann Kapitel 4 auf und stellt das verwendete *State of the Art* Navigationskonzept auf Basis der Error-State Formulierung dar. Hier wurde auf eine strenge Modellierung der Hebelarmeffekte geachtet. Darüber hinaus werden Aspekte betrachtet, die für die Navigation von Fluggeräten von Bedeutung sind.

Zentral ist das Kapitel 5, hier werden zuerst die Simplex Grundlagen betrachtet, um dann daraus das vorgeschlagene Simplex Kalman-Filter herzuleiten. Darüber hinaus wird in Abschnitt 5.3.1 das Verfahren der binären Raumteilung (BSP) dargestellt und gezeigt, wie es in das Simplex Kalman-Filter integriert werden kann, um so beliebige Raumgeometrien als zusätzlich einschränkende Bedingungen in die Schätzung aufzunehmen. Nach den theoretischen Überlegungen wird die Verarbeitung von synthetischen Messungen aus einer Simulationsumgebung betrachtet, anschließend wird das Verfahren mit Realdaten analysiert und einem normalen Kalman-Filter gegenübergestellt. Ebenfalls werden die Einflüsse der binären Raumteilung auf die Wahrscheinlichkeitsdichtefunktion betrachtet.

Während Kapitel 5 nicht festgelegt ist auf die Navigation von Fluggeräten, wird in Kapitel 6 vertieft auf die Aspekte von bemannten Multirotor-Fluggeräten eingegangen. Hervorzuheben ist hier Abschnitt 6.7. Hier wird gezeigt, wie der Simplex Algorithmus mit seinen $\mathscr{L}_1$ Eigenschaften für die Flugregelung von verteilten Systemen genutzt werden kann. Dieses Verfahren wird auf Basis der in Abschnitt

6.2 u. 6.3 vorgestellten Modellierung validiert. In einen redundanten Aufbau von drei Flugsteuerungen werden künstlich falsche Motorstellgrößen induziert, ohne dass dies den sicheren Flug beeinträchtigt.

1.3 Multirotor-Fluggeräte

Die Idee des Helikopters als bemanntes Fluggerät wurde zuerst von Leonardo da Vinci im Jahr 1483 beschrieben (Leishman, 2006), (Bittner, 2014). Da Vinci nannte seine Erfindung *Hélix Pterón*, von den griechischen Wörtern für *Spirale* und *Flügel*, also Schraubenflügler[3]. Daher der Ursprung der Bezeichnung Helikopter, ein Fluggerät, das von Bittner (2014) durch folgende Eigenschaften definiert wird:

- Fluggerät ist schwerer als Luft
- Wird durch einen Motor angetrieben
- Kann mindestens einen Menschen tragen
- Fluggerät kann senkrecht starten und landen
- Schwebeflüge und Transitionen in alle Richtungen sind möglich

Während die Gebrüder Wright bereits im Jahr 1903 mit einem Flugzeug erfolgreich geflogen sind, waren noch etwa drei Jahrzehnte an Forschung und Entwicklung notwendig um einen Helikopter zu bauen, der die beschriebenen Eigenschaften vorweisen konnte. In dieser Zeit gab es eine Vielzahl an Prototypen, die aber alle kaum abheben konnten und nicht gut steuerbar waren. Beispielsweise die vierrotorige Maschine der Gebrüder Breguet oder der Drehflügler von Paul Cornu (Abbildung 1.2). Leishman (2006) nennt als Schwierigkeiten die folgenden Punkte: die Motoren waren nicht leistungsfähig genug, das Verständnis über den Zusammenhang von Leistungsbedarf und erzeugtem Schub fehlte (Gleichung 1.1), die Strukturgewichte waren zu hoch, der Drehmomentausgleich über einen Heckrotor war schwierig umzusetzen oder wurde gar ignoriert, Flugstabilität und Steuerbarkeit waren schwierig, Vibrationen waren die Quelle von unzähligen mechanischen Problemen und Ausfällen und die Fähigkeit zur Autorotation wurde nicht genutzt. Die Maschine von Cornu war aufgrund von Gleichung 1.1 nicht flugfähig und Bittner (2014) stellt auch ein kurzes Abheben in Frage. Allgemein lässt sich für einen idealen und verlustfreien Rotor der Leistungsbedarf (P in Watt) abhängig vom gewünschten Schub (T in Newton), der Luftdichte (ρ in kg/m^3) und der effektiven Gesamtstrahlfläche (A in m^2) berechnen (Bittner, 2014):

$$P = \sqrt{\frac{T^3}{2 \cdot \rho \cdot A}}. \tag{1.1}$$

Als das erste halbwegs flugfähige Multirotor-Fluggerät kann die 4-rotorige Konstruktion des Russen Georges de Bothezat im Jahr 1922 angesehen werden. De Bothezat war auch einer der ersten, der zu dem Thema publizierte[4]. Abbildung 1.3 zeigt die Maschine im Landeanflug. Die Entwicklung erfolgte im Rahmen eines US Army Forschungsprojekts; das Projekt wurde aber aufgrund der hohen Kosten und

[3] Der Begriff Multikopter bzw. Kopter ist daher eigentlich falsch, da hier Teile von *Hélix* bzw. dem Genitiv *Héliko* mit dem Wort *Pterón* vermischt werden.

[4] de Bothezat, G. 1919. *The General Theory of Blade Screws*, NACA TR 29

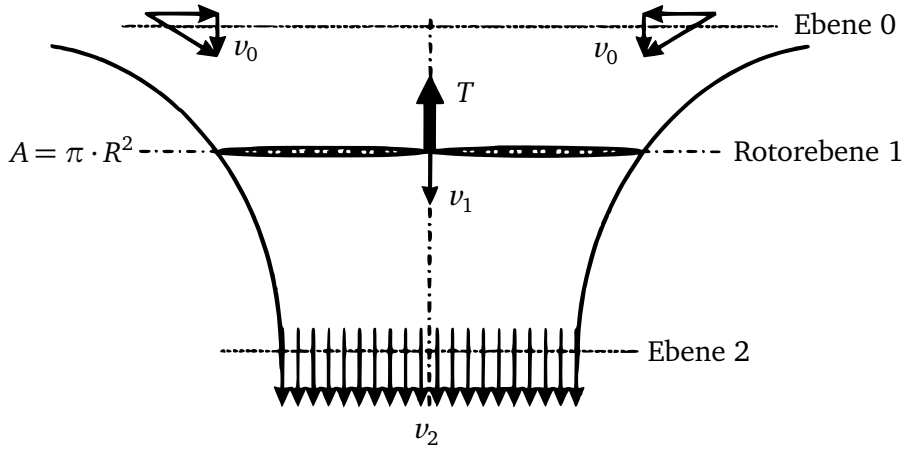


Abbildung 1.1.: Grundprinzip eines „idealen" Rotors, nach einer Abbildung von Bittner (2014).

der eher schlechten Flugeigenschaften wieder beendet (Leishman, 2006). Etwa zur gleichen Zeit entwickelte der Franzose Étienne Œhmichen ebenfalls einen 4-rotorigen Helikopter (Abbildung 1.4) und konnte damit einige Weltrekorde bei der Fédération Aéronautique Internationale (FAI) aufstellen, u. a. den ersten Flug mit einem Helikopter über 360 m am 14. April 1924 und das Anheben von 200 kg Nutzlast am 14. Sept. 1924. Auch Œhmichens Fluggerät konnte sich nicht durchsetzen. Dennoch können De Bothezat und Œhmichen als die ersten Entwickler eines bemannten Multikopters angesehen werden. Letztendlich konnte sich die Helikopterkonstruktion mit einem Hauptrotor und einem Heckrotor zum Drehmomentausgleich Ende der 1930er Jahre durchsetzen. Hier sind nach Leishman (2000) besonders die Entwicklungen von Louis Breguet und Rene Dorand zu erwähnen, sowie der Typ VS-300 von Igor Sikorsky. Abbildung 1.1 zeigt das grundsätzliche Prinzip um Auftrieb für ein Multirotor-Fluggerät zu

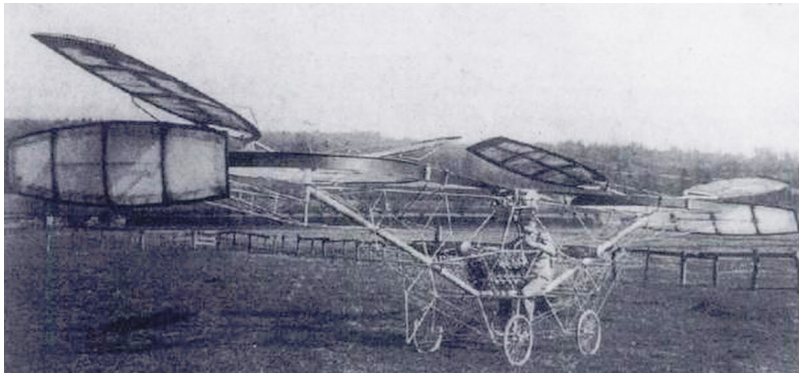


Abbildung 1.2.: Multirotor-Fluggerät von Paul Cornu (1907), Abbildung aus Leishman (2006).

erzeugen, zurückgehend auf die Erkenntnisse von Bernoulli. Luft wird oberhalb des Rotors mit der Geschwindigkeit v_0 angezogen, beschleunigt und durchströmt den Rotor mit der Geschwindigkeit v_1. Dabei entsteht ein Druckunterschied, der den gewünschten Auftrieb T erzeugt. Der Luftmassendurchsatz ist in diesem Strom immer gleich (Bittner, 2014).

Bei einem gewöhnlichen Multikopter mit fest verbauten Antriebseinheiten handelt es sich um ein unteraktuiertes System, das heißt, dass die 3 translatorischen und 3 rotatorischen Freiheitsgrade nicht unabhängig voneinander eingestellt werden (De Monte, 2015). Der Vorteil davon ist, dass man mecha-

Abbildung 1.3.: Flug des de Bothezat Multirotor-Fluggeräts in Ohio, USA (1923). Geflogen von Major Bane, US Army. Abbildungsquelle: Smithsonian National Air and Space Museum.

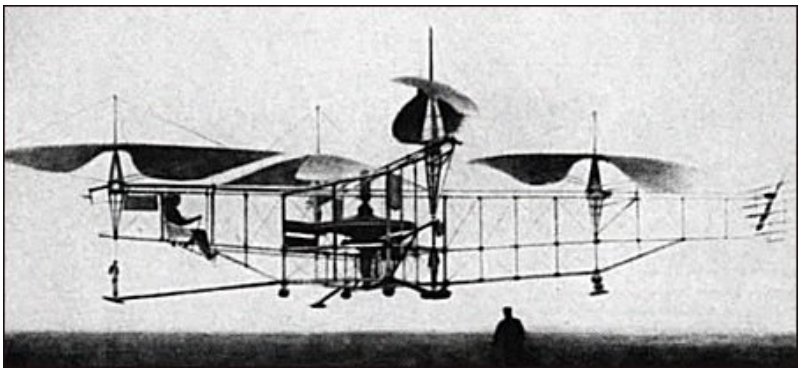


Abbildung 1.4.: Flug des L'hélicoptère N°2 d'Étienne Œhmichen.

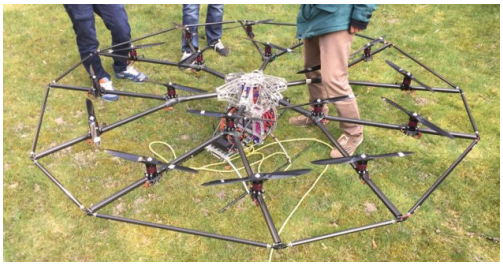
nische Komplexität gegen Komplexität in der Software eintauscht, hauptsächlich im Bereich Zustandsschätzung und Regelungsalgorithmen. Das entspricht einer logischen Entwicklung: Algorithmen, Sensoren und Rechner werden zuverlässiger, leistungsstärker und billiger. Der initiale Aufwand ist allerdings signifikant höher, da Software für die Luftfahrt nach strengen Vorgaben zu entwickeln ist, üblicherweise nach der Richtlinie DO-178C. Die Richtlinie stellt auch keine endgültige Garantie dar, dass die Software fehlerfrei ist. Es ist durchaus möglich, dass falsche Annahmen in die Entwicklungsprozesse eingeflossen sind.

Dafür kann z. B. die teure und wartungsintensive Mechanik eines Hubschraubers eingespart werden, da die entsprechenden Bauteilgruppen bei einem Multikopter schlicht nicht vorhanden sind. Zufällige oder systematische mechanische Fehler werden vermieden.

Bei dem angesprochenen Mehraufwand in der Algorithmenentwicklung entstehen gleichzeitig aber auch Systeme, die den Piloten entlasten können. Etwa 60-80 % der Flugunfälle lassen sich auf menschliche Fehler zurückführen (Shappell et al., 2017). Die Ursache für das menschliche Versagen ist ein Thema für sich und nicht alle Fehler lassen sich mit besseren Sensoren und Algorithmen vermeiden, aber nach Shappell et al. (2017) spielt immerhin bei mehr als der Hälfte (56.5 %) der Zwischenfälle das Geschick des Piloten eine signifikante Rolle.

Es gibt auch Ansätze, bei denen die Antriebseinheiten mit steuerbarem Tiltwinkel ausgestattet sind. Z. B. wird ein entsprechendes Quadrotor UAV von Ryll et al. (2013) untersucht, sowie an der ETH Zü-

rich wurde an einem Fluggerät mit 12 Aktuatoren geforscht (Elkhatib, 2017), derartig Systeme werden im Weiteren nicht betrachtet. Typische Systeme für die folgenden Betrachtungen sind in den Abbildungen 1.5a – 1.5d zu sehen. Davon ist der Volocopter[5] VC200 in Abb. 1.5d ein manntragendes Multirotor VTOL Fluggerät. Die weiteren Test UAVs wurden an der Hochschule Karlsruhe (HSKA) entwickelt. Abb. 1.5c zeigt ein Multirotor UAV mit sechs Antrieben, bestückt mit einer Navka FC4 Einheit (siehe Abschnitt 5.4.1). Algorithmen mit Forschungscharakter sollten, nach entsprechenden Software-In-The-Loop und Hardware-In-The-Loop Tests, zuerst auf Systemen mit geringer Abflugmasse getestet werden. Tests, die beispielsweise die Motorallokation (betrachtet in Abschnitt 6.6) betreffen, können auf den Systemen in Abbildung 1.5b und 1.5a getestet werden, sie verfügen beide über 18 Motoren. Der VC25A wird über zwei Netzteile mit Strom versorgt und wurde u. A. für Langzeitflugtests (länger als 1 Stunde) eingesetzt.



(a) *VC25C* UAV (25 kg TOW).

(b) *VC25A* UAV mit Netzstromversorgung (25 kg TOW).

(c) *Navka* Hexakopter (ca. 2 kg TOW).

(d) Autor steuert den VC200 (450 kg TOW).

Abbildung 1.5.: Modelle eines manntragenden Multikopters für die Validierung der Navigations- und Regelalgorithmen (TOW = Take-Off Weight, bzw. max. Abfluggewicht).

[5] Volocopter GmbH: `http://www.volocopter.de`

2 Parameter- und Zustandsschätzung

2.1 Rekursive Zustandsschätzung und Steuerung auf stochastischer Grundlage

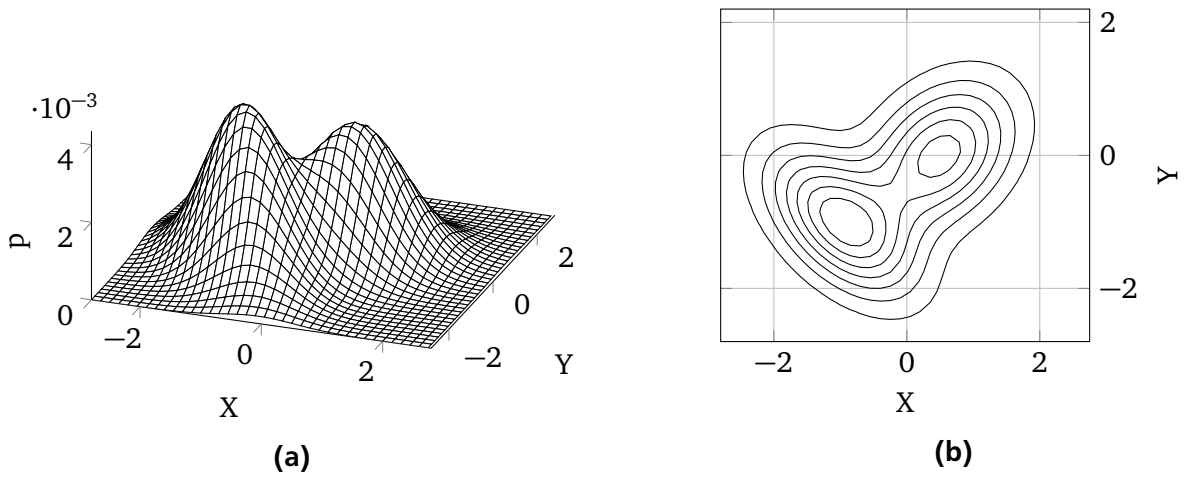


Abbildung 2.1.: Beispielhafte Darstellung einer zufällig gewählten multimodalen (nicht normalverteilten) Wahrscheinlichkeitsdichtefunktion für einen zweidimensionalen Zustandsvektor.

Kein physikalisches System (wie beispielsweise ein Luftfahrtgerät) lässt sich fehlerfrei als deterministisches System beschreiben. Maybeck (1979) nennt drei Hauptgründe, warum eine *stochastische* Betrachtungsweise notwendig ist:

a) Kein *mathematisches Modell* ist perfekt. Beispielsweise sind physikalische Modelle oft auf Grundlage der klassischen Physik modelliert, relativistische Effekte werden oft ignoriert.

b) *Dynamische Systeme* sind i. A. nicht nur von den eigenen Eingaben gesteuert (beispielsweise durch Pilotenvorgabe) sondern werden durch unbekannte Kräfte beeinflusst. Beispielsweise durch Windböen oder Fertigungstoleranzen in der Aktorik.

c) *Messungen sind nicht fehlerfrei*. Kein Sensor liefert fehlerfrei alle benötigten Informationen. Sensorbeobachtungen können systematisch abweichen und durch *Rauschen* bzw. zufällige Fehler ϵ verfälscht sein.

Diese unvermeidbaren Unsicherheiten führen somit zwingend zu einer Betrachtung der Wahrscheinlichkeiten bei zeitlich ablaufenden Prozessen. Gesucht ist fortlaufend der Zustandsvektor von einem Objekt. Geschätzt werden soll der Zustandsvektors bzw. Navigationszustandsvektor $\mathbf{y}$, der sich von Epoche zu Epoche ändert, gekennzeichnet durch den Index k: $\mathbf{y}_0$, $\mathbf{y}_1$, …, $\mathbf{y}_k$. Ebenso liegen über die Zeit hinweg Messungen vor $\mathbf{L}_k = \{\mathbf{l}_0, \mathbf{l}_1, \ldots, \mathbf{l}_k\}$, sowie Steuer- bzw. *Control*-Eingaben[1] $\mathbf{U}_k = \{\mathbf{u}_0, \mathbf{u}_1, \ldots, \mathbf{u}_k\}$. Die Messungen $\mathbf{l}$ lassen direkt oder indirekt Rückschlüsse auf den gesuchten Zustandsvektor $\mathbf{y}$ zu. Der

[1] Zu den Steuer- bzw. Control-Eingaben zählen nach Thrun et al. (2005) auch beispielsweise Rad-Odometriemessungen, da sie den Effekt eines Control-Inputs beschreiben.

Control-Vektor $\mathbf{u}$ hingegen erlaubt eine Vorhersage, wie sich ein Zustandsvektor zeitlich gesehen entwickeln wird. Abbildung 2.2 stellt die zeitliche Abhängigkeit dar. Der Zustand zum Zeitpunkt k hängt stochastisch vom vorhergehenden Zustand zum Zeitpunkt $k-1$ ab, sowie der Steuereingabe aus $k-1$. Die Messungen zum Zeitpunkt k sind allein abhängig vom Zustand zum Zeitpunkt k. Man spricht von einem *Hidden Markov Model* (HMM) (Thrun et al., 2005). Bevor in den folgenden Kapiteln der Blick auf spezielle Zustandsschätzer mit gewissen Annahmen gerichtet wird, soll in diesem Abschnitt das allgemeingültige Fundament dafür betrachtet werden.

In dem Beitrag von Jäger (2018) wird beispielsweise darauf hingewiesen, dass die Verfahren der Zustandsschätzung von dynamischen Zuständen auf einer gemeinsamen Basis aufgebaut sind: Markov-Reihen und deren Überführung in Bayes-Schätzungen. Dieser probabilistische Ansatz wurde insbesondere durch Beiträge im Bereich der Robotik populär (Burgard et al., 1996), (Thrun, 1998), (Dellaert et al., 1999), (Fox et al., 2001), (Thrun et al., 2005).

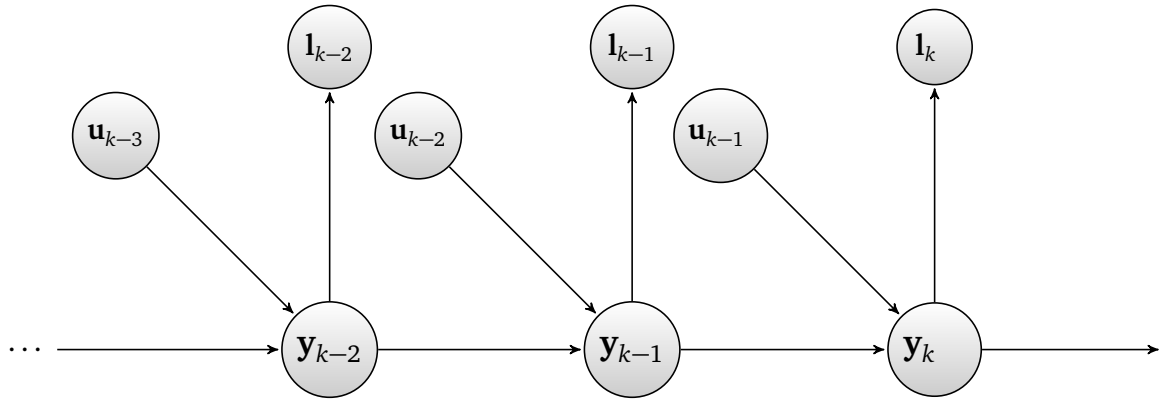


Abbildung 2.2.: Probabilistische Zustandsschätzung in Graphendarstellung. Nach einer Abbildung von Thrun et al. (2005).

Die Grundlage bildet dabei das notwendige Betrachten von Wahrscheinlichkeitsdichtefunktionen. Für einen Zustandsvektor $\mathbf{y}$ sei eine Wahrscheinlichkeitsdichtefunktion $p(\mathbf{y})$ gegeben, die angibt, wie wahrscheinlich es ist, dass die stochastische Größe $\mathbf{y}$ einen bestimmten Wert einnimmt. Im engl. *probability density function* (PDF) genannt. Die Wahrscheinlichkeit, dass zwei unabhängige Variablen eine bestimmte Wertekombination einnehmen lässt sich aus dem Produkt der Einzelwahrscheinlichkeiten berechnen:

$$p(A,B) = p(A) \cdot p(B). \tag{2.1}$$

Beispielhaft zeigt Abbildung 2.1 eine zufällig gewählte PDF für einen Zustandsvektor $\mathbf{y}$ mit zwei Komponenten: X u. Y. Es wird im Folgenden von beliebigen Verteilungen ausgegangen. Allgemein gilt: Das Integral über die Dichtefunktion p hinweg muss 1 ergeben:

$$\int p(\mathbf{y})\, \mathrm{d}y = 1. \tag{2.2}$$

Der Erwartungswert des Vektors $\mathbf{y}$ errechnet sich aus der Summe aller möglichen Ausprägungen, gewichtet mit der jeweiligen Wahrscheinlichkeit dieser Ausprägung, siehe z. B. Jäger et al. (2005):

$$E[\mathbf{y}] = \int \mathbf{y} \cdot p(\mathbf{y}) \, \mathrm{d}y \tag{2.3}$$

Einen sehr wichtigen Zusammenhang zwischen Wahrscheinlichkeiten gibt der Satz von Bayes[2]. Dieser Satz beschreibt die *bedingte* Wahrscheinlichkeit $p(A \mid B)$, also jene Wahrscheinlichkeit, dass A zutrifft, unter der Annahme, dass B eingetreten ist:

$$p(A \mid B) = \frac{p(B \mid A) \cdot p(A)}{p(B)}. \tag{2.4}$$

$p(A)$ und $p(B)$ sind die a priori Wahrscheinlichkeiten. Sind die Variablen A und B stochastisch unabhängig, so vereinfacht sich der Zusammenhang:

$$p(A \mid B) = \frac{p(B \mid A) \cdot p(A)}{p(B)} = \frac{p(B) \cdot p(A)}{p(B)} = p(A). \tag{2.5}$$

Für die Zustandsschätzung ist die Dichtefunktion $p(\mathbf{y})$ gesucht, unter der Berücksichtigung aller Messungen ($\mathbf{L}_k$) bis zum Zeitpunkt k (Dellaert et al., 1999):

$$p(\mathbf{y}_k \mid \mathbf{L}_k). \tag{2.6}$$

Diese Bestimmung erfolgt über die beiden folgenden Operationen

- Vorhersage (Behandlung der *State Transition Probability*) und
- Korrektur (Behandlung der *Measurement Probability*).

Vorhersageschritt: Die Dichtefunktion der Vorhersage beinhaltet alle bisherigen Informationen für den Zeitpunkt k, bis auf die Messungen zum Zeitpunkt k. Dabei nimmt man an, dass der aktuelle Zustand $\mathbf{y}_k$ nur von dem bisherigen Zustand $\mathbf{y}_{k-1}$ und dem bekannten Control-Input aus der vorhergehenden Epoche $\mathbf{u}_{k-1}$ abhängt[3]. Man nimmt also damit an, dass es sich um eine *Markov-Reihe* handelt (Dellaert et al., 1999). Die Vorhersage auf Basis von $\mathbf{u}_{k-1}$ lässt sich selbst als bedingte Wahrscheinlichkeit ausdrücken: $p(\mathbf{y}_k \mid \mathbf{y}_{k-1}, \mathbf{u}_{k-1})$. Über die Chapman-Kolmogorov Gleichung (2.7), kann dann die a priori Dichte für $\mathbf{y}_k$ berechnet werden (Wendel, 2011):

$$p(\mathbf{y}_k \mid \mathbf{L}_{k-1}) = \int p(\mathbf{y}_k \mid \mathbf{y}_{k-1}, \mathbf{u}_{k-1}) \cdot p(\mathbf{y}_{k-1} \mid \mathbf{L}_{k-1}, \mathbf{U}_{k-2}) \, \mathrm{d}\mathbf{y}_{k-1}. \tag{2.7}$$

[2] Eine Herleitung ist in Maybeck (1979) zu finden.

[3] In manchen Quellen ist der Control-Input, der zu dem aktuellen Zustand führt mit dem Index k versehen und nicht mit $k-1$.

Korrekturschritt: Unter der weiteren Markov-Annahme, dass die zum Zeitpunkt k vorliegenden Messungen gen unabhängig sind von vergangenen Messungen ($\mathbf{L}_{k-1}$), so lässt sich die gesuchte a posteriori Dichte mit Hilfe der Formel 2.4 von Bayes bestimmen (Dellaert et al., 1999):

$$p(\mathbf{y}_k \mid \mathbf{L}_k) = \frac{p(\mathbf{l}_k \mid \mathbf{y}_k) \cdot p(\mathbf{y}_k \mid \mathbf{L}_{k-1})}{p(\mathbf{l}_k \mid \mathbf{L}_{k-1})} \tag{2.8}$$

Die bedingte Wahrscheinlichkeit $p(\mathbf{l}_k \mid \mathbf{y}_k)$ wird Messwahrscheinlichkeit bzw. *measurement probability* genannt. Da der Nenner von Gl. 2.8 nicht von $\mathbf{y}$ abhängig ist, wird er der Übersichtlichkeit wegen von Thrun et al. (2005) zu einem normalisierenden Faktor η zusammengefasst:

$$p(\mathbf{y}_k \mid \mathbf{L}_k) = \eta \cdot p(\mathbf{l}_k \mid \mathbf{y}_k) \cdot p(\mathbf{y}_k \mid \mathbf{L}_{k-1}). \tag{2.9}$$

Thrun et al. (2005) führen das Konzept der *Belief* Funktion „$bel(\ldots)$“ ein. Der Belief ist die bedingte Wahrscheinlichkeitsdichte für den aktuell geschätzten a posteriori Zustand, inklusive der Steuereingaben, die zu dem aktuellen Zustand geführt haben (vgl. Gl. 2.9):

$$bel(\mathbf{y}_k)^+ = p(\mathbf{y}_k \mid \mathbf{y}_0, \mathbf{L}_k, \mathbf{U}_{k-1}). \tag{2.10}$$

Unter der Markov-Annahme, dass der Systemzustand die Zusammenhänge komplett beschreibt und somit vorhergehende Ereignisse keinen Einfluss mehr haben:

$$bel(\mathbf{y}_k)^+ = \eta \cdot p(\mathbf{l}_k \mid \mathbf{y}_k) \cdot bel(\mathbf{y}_k)^-. \tag{2.11}$$

Gleichung 2.11 enthält den a priori Belief (vgl. 2.7), also der *Belief*, der noch keine Kenntnis über die Messungen der aktuellen Epoche $\mathbf{l}_k$ hat:

$$bel(\mathbf{y}_k)^- = \int p(\mathbf{y}_k \mid \mathbf{y}_{k-1}, \mathbf{u}_{k-1}) \cdot bel(\mathbf{y}_{k-1})^+ \, \mathrm{d}\mathbf{y}_{k-1}. \tag{2.12}$$

Das hochgestellte $^+$ kennzeichnet den a posteriori Belief, ein hochgestelltes $^-$ den a priori Belief. Die Gleichungen 2.11 und 2.12 bilden damit die Grundlage der rekursiven ***Bayes Filterung***. Die Bayes Filterung benötigt einen „Anfangsbelief“. Dies kann beispielsweise eine Gleichverteilung sein. Die Bayes Filterung bildet das Fundament der rekursiven Zustandsschätzung bzw. der Navigationszustandsschätzung. Jedoch ist eine rein analytische Lösung für die Bayes Filterung in den meisten Fällen nicht möglich (Thrun et al., 2005), (Wendel, 2011). In Abschnitt 2.2 werden Sonderfälle der Bayes Filterung betrachtet, die für praktische Anwendungen relevant sind.

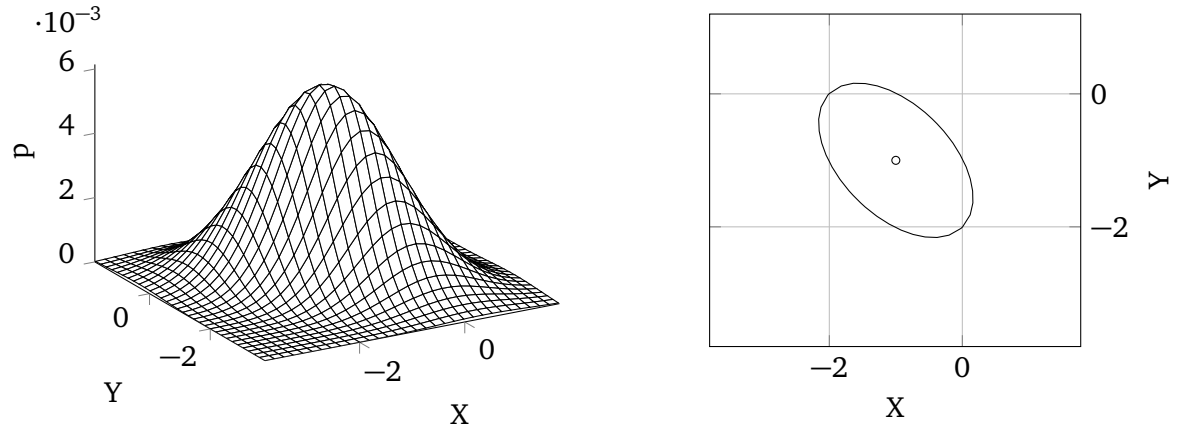


Abbildung 2.3.: Beispielhafte Darstellung einer normalverteilten Wahrscheinlichkeitsdichtefunktion für einen zweidimensionalen Zustandsvektor.

2.2 Sonderfälle

Aufbauend auf den theoretischen Grundlagen der probabilistischen Zustandsschätzung in Abschnitt 2.1 sollen in den folgenden Abschnitten Lösungsverfahren bzw. Lösungsverfahren mit Einschränkungen betrachtet werden.

In Abschnitt 2.3 werden die Partikelfilter als eine mögliche[4] allgemeingültige numerische Lösung für die Gleichungen der stochastischen Filterung (Bayes-Filterung) vorgestellt. Partikelfilter erlauben die Behandlung von beliebigen Wahrscheinlichkeitsdichtefunktionen.

Eine Sonderstellung hat die Gauß-Verteilung bzw. Normalverteilung. In Abhängigkeit der Kovarianzmatrix $\mathbf{Q}_{yy}$ für einen Erwartungswert $\boldsymbol{\mu}$ lautet die Dichtefunktion der n-dimensionalen multivariaten Normalverteilung (Jäger et al., 2005):

$$p(\mathbf{y}) = \frac{1}{\left(\sqrt{2\cdot\pi}\right)^n \cdot \sqrt{\det\left(\mathbf{Q}_{yy}\right)}} \cdot \exp\left\{ -\frac{1}{2}(\mathbf{y}-\boldsymbol{\mu})^T \cdot \mathbf{Q}_{yy}^{-1} \cdot (\mathbf{y}-\boldsymbol{\mu}) \right\}. \tag{2.13}$$

Bei vorliegender Normalverteilung kann also die Wahrscheinlichkeitsdichtefunktion über den Erwartungswert und die Kovarianzmatrix beschrieben werden. Werden n Variablen ξ_i mit jeweils *beliebiger* Verteilung aufsummiert, so nähert sich die Verteilung dieser Summe asymptotisch einer Normalverteilung an (Cramér, 1946):

$$\xi = \xi_1 + \xi_2 + \ldots + \xi_n. \tag{2.14}$$

Für große Werte von n ist ξ näherungsweise normalverteilt um den Mittelwert m mit der Standardabweichung σ: $\xi \sim \mathcal{N}(m,\sigma)$. Dieser fundamentale Satz ist als *zentraler Grenzwertsatz* bekannt (Cramér, 1946). Der zentrale Grenzwertsatz erklärt damit, warum die Normalverteilung so häufig anzutreffen ist.

[4] Für weitere Verfahren sei auf Thrun et al. (2005) verwiesen.

Kann das zeitliche Verhalten eines Systems rein über lineare Zusammenhänge beschrieben werden und sind darüber hinaus alle einwirkenden Messungen normalverteilt, so stellt das in Abschnitt 2.8 betrachtete Kalman-Filter ein optimales Verfahren dar. Metzger (2006) zeigt, dass sich unter diesen Annahmen das Kalman-Filter aus den Gleichungen der Bayes-Filterung herleiten lässt.

2.3 Partikelfilter

Einen allgemeingültigen numerischen Lösungsansatz für die Bayes Filterung stellt die Monte-Carlo Methode dar, ursprünglich von Gordon et al. (1993) vorgeschlagen. Beliebige Verteilungen lassen sich mit sogenannten *Partikeln* abbilden, die jeweils eine mögliche Ausprägung eines Zustandsvektors $\mathbf{y}$ darstellen und die Verteilung bzw. den *Belief* somit über eine diskrete Instanziierung nachbilden. Desto mehr Partikel sich an einer Stelle gruppieren, desto höher die Wahrscheinlichkeit, dass diese Partikel den gesuchten Zustand repräsentieren (Thrun et al., 2005), (Stachniss und Burgard, 2014):

$$\mathbf{y}_k^i \sim bel\,(\mathbf{y}_k)^+ . \tag{2.15}$$

Für N Partikel, jeweils mit dem Index i gekennzeichnet, ergibt sich die Menge der Partikel $\mathbf{Y}$ zur Epoche k:

$$\mathbf{Y}_k = \{\mathbf{y}_k^1, w_k^1, \mathbf{y}_k^2, w_k^2, \ldots, \mathbf{y}_k^N, w_k^N\}. \tag{2.16}$$

Jedes Partikel entspricht einer möglichen Ausprägung des Zustandsvektors. Zusätzlich erhält jedes Partikel ein Gewicht w^i. Die Summe der Gewichte muss genau 1 ergeben, vergleichbar mit dem Integral über die zu approximierende Dichtefunktion p:

$$\sum_{i=1}^{N} w^i = 1. \tag{2.17}$$

Die Anfangsverteilung ist je nach Anwendung unterschiedlich. Häufig wird eine Gleichverteilung genutzt. Aber u. U. liegen für die erste Epoche Informationen vor, die genutzt werden können. Wichtig ist, dass die Komponenten von $\mathbf{y}_i$ über den Bereich hinweg verteilt sind, der realistischerweise auftreten kann. Die Wahrscheinlichkeitsdichtefunktion wird formal definiert mit Hilfe der Dirac δ Verteilungsfunktion:

$$p(\mathbf{y}_k) = \sum_{i=1}^{N} w_k^i \cdot \delta(\mathbf{y}_k - \mathbf{y}_k^i). \tag{2.18}$$

Womit ausgedrückt wird, dass je höher die Summe der Partikelgewichte, die in eine Region fallen, desto höher ist die Wahrscheinlichkeit, dass der wahre Zustand in dieser Region liegt (Stachniss und Burgard, 2014). Die Dirac δ Verteilungsfunktion ist mit folgenden Eigenschaften definiert:

$$\delta(\mathbf{y}) = 0 \text{ für } \mathbf{y} \neq 0 \text{ und} \tag{2.19}$$

$$\int_{-\infty}^{\infty} \delta\mathbf{y}\, \mathrm{d}y = 1. \tag{2.20}$$

Vorhersageschritt: Für den Vorhersageschritt wird ein neuer Satz Partikel für die Epoche k verteilt; basierend auf dem Control-Input $\mathbf{u}_{k-1}$ sowie der dazugehörigen Zustandsübergangsdichtefunktion $p(\mathbf{y}_k \mid \mathbf{y}_{k-1}, \mathbf{u}_{k-1})$. Der Control-Input liegt in der Epoche $k-1$ vor und führt zu einem neuen Zustand in der Epoche k. Der Vektor $\mathbf{u}_{k-1}$ muss nicht zwangsweise eine Stellgröße sein. $\mathbf{u}_{k-1}$ kann oft auch eine Sensormessung enthalten, die direkt die Auswirkung von gesetzten Stellgrößen beschreibt (Thrun et al., 2005, Kap. 5.4.1). Die neue Menge an Partikeln muss also so verteilt sein, wie es die Zustandsübergangsdichtefunktion erwarten lässt. In der Praxis werden häufig nicht eine neue Menge an Partikeln $\mathbf{Y}_k$ gezogen, stattdessen werden die Partikel $\mathbf{Y}_{k-1}$ entsprechend einer Vorhersagefunktion f prädiziert (Wendel, 2011):

$$\mathbf{y}_k^i = f\left(\mathbf{y}_{k-1}^i, \mathbf{u}_{k-1}\right) + \mathbf{w}. \tag{2.21}$$

Der Vektor $\mathbf{w}$ modelliert dabei für jedes Partikel die Unsicherheit in der Vorhersage und muss konsistent sein mit der Zustandsübergangsdichtefunktion, sodass für die Verteilung der Partikel gilt:

$$\mathbf{y}_k^i \sim bel\left(\mathbf{y}_k\right)^-, \tag{2.22}$$

also als numerische Lösung von Gl. 2.12

$$bel(\mathbf{y}_k)^- = \int p(\mathbf{y}_k \mid \mathbf{y}_{k-1}, \mathbf{u}_{k-1}) \cdot bel(\mathbf{y}_{k-1})^+ \, \mathrm{d}\mathbf{y}_{k-1}. \tag{2.23}$$

Korrekturschritt und Resampling: Das Ziel der Korrektur ist es, dass die Partikel entsprechend

$$\mathbf{y}_k^i \sim bel\left(\mathbf{y}_k\right)^+ \tag{2.24}$$

verteilt sind. Analog zur Vorhersage die Lösung von Gl. 2.11

$$bel(\mathbf{y}_k)^+ = \eta \cdot p(\mathbf{l}_k \mid \mathbf{y}_k) \cdot bel(\mathbf{y}_k)^-. \tag{2.25}$$

Für jede vorliegende Messung $\mathbf{l}_k$ wird nun das Gewicht der Partikel entsprechend der Dichtefunktion der Messungen $p(\mathbf{l}_k \mid \mathbf{y}_k^i)$ gesetzt:

$$w^i =: p(\mathbf{v}^i) \cdot w^i, \tag{2.26}$$

mit dem Residuenvektor $\mathbf{v}^i$, der den Widerspruch zwischen der vorliegenden Messung $\mathbf{l}_k$ und der erwarteten Messung für das i-te Partikel darstellt:

$$\mathbf{v}^i = \mathbf{l}_k - h\left(\mathbf{y}^\mathbf{i}\right). \tag{2.27}$$

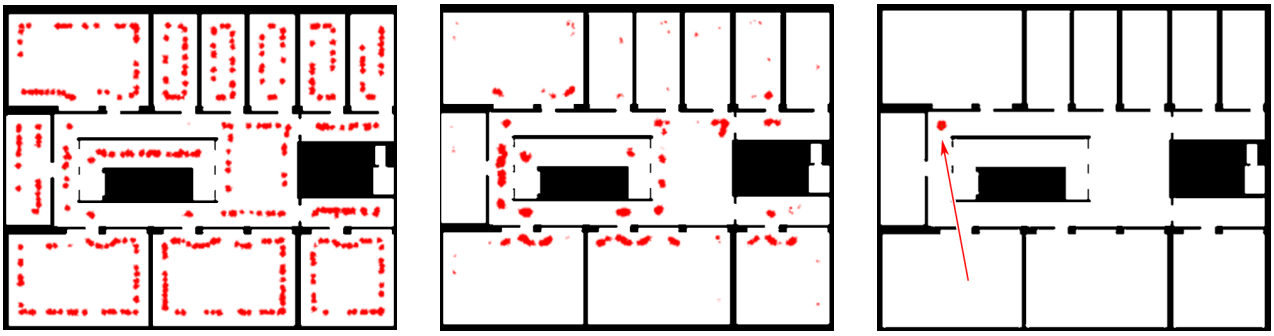
Hier liegt die Stärke des Partikelfilters, da beliebige Beobachtungsgleichungen mit beliebiger Dichtefunktion p eingebunden werden können. Um den Korrekturschritt abzuschließen muss zuletzt noch das sogenannte Resampling stattfinden; dieser Schritt ist bei der Bayes-Filterung nicht notwendig. Vor dem Resampling ist sicherzustellen, dass die Gewichte normalisiert sind:

$$w^j \leftarrow \frac{w^j}{\sum_{i=1}^{N} w^i} \tag{2.28}$$

Thrun et al. (2005) nennen das Resampling auch den „*Trick*“ hinter dem Partikelfilter. Im Resampling Schritt werden N Partikel aus der Menge $\mathbf{Y}_k$ gezogen. Die Wahrscheinlichkeit, dass ein Partikel $\mathbf{y}_k^i$ zur neuen Menge der Partikel gehört, ist proportional zu dem zugehörigen Gewicht w_k^i. Somit werden Partikel mit einem geringen Gewicht tendenziell verworfen. Während Partikel mit hohem Gewicht nach dem Resampling zunächst mehrfach vorhanden sind. Stachniss und Burgard (2014) nennen es auch „*survival of the fittest*“. Nach der Neuverteilung werden alle Gewichte w^i auf den einheitlichen Wert $1/N$ gesetzt. Vor dem Resampling sind die Partikel näherungsweise nach $bel\,(\mathbf{y}_k)^-$ verteilt, danach entsprechend $bel\,(\mathbf{y}_k)^+$ (Thrun et al., 2005). Bolić et al. (2004) oder Gustafsson (2010) zeigen wie sich das Resampling programmiertechnisch effizient umsetzen lässt.

Beispiel: In Abbildung 2.4 werden drei Epochen einer Partikelfilter Zustandsschätzung gezeigt; jeder Punkt stellt eine mögliche Position auf demselben Stockwerk dar. Die eingezeichneten Wände schränken den Lösungsraum ein und sind mit den Ungleichungen aus dem BSP Verfahren vergleichbar (siehe Abschnitt 5.3.2). Es darf sich kein Partikel in den Wänden befinden. Partikel werden prädiziert und Messungen gewichten die Partikel dann entsprechend der Übereinstimmung mit der Dichtefunktion der Messungen. In diesem Fall sind das Laserdistanzmessungen zu den Wänden mit normalverteilten Residuen. Der Zustandsvektor schätzt die 2D Position und Geschwindigkeit und es wird angenommen, dass sich das Objekt mit konstanter Geschwindigkeit fortbewegt.

In der 0-ten Epoche wird mit einer Gleichverteilung begonnen. Anhand der Gewichte werden die Partikel nach einem Resamplingschritt wieder neu verteilt, bis in der letzten Epoche (Abb. 2.4c) keine multimodale Verteilung mehr vorliegt. Partikelfilter approximieren die Zusammenhänge der Bayes Filterung (siehe Gl. 2.11 und 2.12) durch die hohe Anzahl an Partikeln. Die Anzahl der benötigten Partikel



(a) Epoche 1. **(b)** Epoche 2. **(c)** Epoche 3.

Abbildung 2.4.: Drei Epochen einer Partikelfilterzustandsschätzung im Gebäude B auf dem Campus der Hochschule Karlsruhe (HSKA).

skaliert exponentiell mit der Dimension des Zustandsvektors, was zu Limitierungen führen kann (Thrun, 2002).

2.4 Methode der kleinsten Quadrate

Die Ausgleichungsrechnung nach der Methode der kleinsten Quadrate ist seit Gauss fester Bestandteil im Werkzeugkasten aller Naturwissenschaften; im Bereich der Geodäsie besonders für die Verarbeitung von fehlerbehafteten Sensordaten sowie ganz allgemein zur Auswertung von raumbezogenen Daten. Üblicherweise überbestimmte Probleme erlauben eine tiefgreifende Bewertung: Zustands- und Parameterschätzung, Behandlung von zeitlich ablaufenden Prozessen, statistische Genauigkeitsmaße, Fehlerfortpflanzung, Analyse der Fehlerverteilung und Korrelationen, Restriktionen bzw. Ungleichungen, Signifikanztests und besonders die Behandlung von zufälligen und systematischen Messunsicherheiten.

Die grundlegende Annahme ist, dass sich Beobachtungen ($\mathbf{l}$) als Funktion der gesuchten Größen ($\mathbf{y}$) ausdrücken lassen. Man spricht von der Ausgleichung nach vermittelnden Beobachtungen bzw. einem Gauss-Markov Modell (GMM). Für die wahren Größen (gekennzeichnet mit dem Diakritikum $\sim$) und den wahren Fehlern $\boldsymbol{\epsilon}$ gilt:

$$\tilde{\mathbf{l}} = f\left(\tilde{\mathbf{y}}\right) \tag{2.29}$$

$$\mathbf{l} - \boldsymbol{\epsilon} = f\left(\tilde{\mathbf{y}}\right). \tag{2.30}$$

Bei fehlerbehafteten Beobachtungen $\mathbf{l}$ können die wahren Werte oftmals nicht bestimmt werden. Die tatsächlichen Beobachtungen lassen jedoch über die Funktion f eine *Schätzung* der Parameter $\hat{\mathbf{y}}$ zu:

$$\mathbf{l} + \mathbf{v} = f\left(\hat{\mathbf{y}}\right). \tag{2.31}$$

Je nach Schätzmethode bleibt ein Residuenvektor $\mathbf{v}$ übrig. Man spricht von *Regression* (lat. regredi: zurückgehen/zurückkommen), man schätzt ausgehend von den Messungen die Parameter. Für Beobachtungen mit normalverteilten Residuen $\mathbf{v} \sim \mathcal{N}(0, \sigma^2)$ ohne systematischen Fehleranteil und linearen funktionalen Zusammenhängen ist die klassische Methode der kleinsten Quadrate die Schätzmethode

mit der besten Effizienz (Jäger et al., 2005). D. h. diese Schätzmethode liefert die genauesten Ergebnisse und keine andere Schätzmethode liefert eine kleinere Varianz der geschätzten Parameter. Man spricht auch vom *best linear unbiased estimator* (BLUE). Die Methode der kleinsten Quadrate ($\mathscr{L}_2$ Norm) minimiert die Summe der quadrierten Residuen, auch Minimierung der Euklidischen Norm genannt:

$$\mathbf{v}^T \cdot \mathbf{v} \rightarrow \text{Min.} \tag{2.32}$$

Bzw. unter Berücksichtigung einer Gewichtsmatrix $\mathbf{P}$:

$$\mathbf{v}^T \cdot \mathbf{P} \cdot \mathbf{v} \rightarrow \text{Min.} \tag{2.33}$$

Üblicherweise leitet sich $\mathbf{P}$ aus der Inversen der Kofaktormatrix $\mathbf{Q}_{ll}$ bzw. der Kovarianzmatrix der Messungen $\mathbf{C}_{ll}$ ab:

$$\mathbf{P} = \mathbf{Q}_{ll}^{-1} = \sigma_0^2 \cdot \mathbf{C}_{ll}^{-1}. \tag{2.34}$$

Die Kofaktormatrix steht über den skalaren A priori-Varianzfaktor σ_0^2 mit der Kovarianzmatrix in Verbindung (Jäger et al., 2005):

$$\mathbf{C}_{ll} = \sigma_0^2 \cdot \mathbf{Q}_{ll}. \tag{2.35}$$

Dies setzt insgesamt voraus, dass sich die Wahrscheinlichkeitsdichtefunktion der Messungen über eine Kovarianzmatrix (siehe Gl. 2.13) beschreiben lässt. Sind lineare Zusammenhänge gegeben, so lässt sich Gleichung 2.31 mit Hilfe einer Designmatrix $\mathbf{A}$ ausdrücken:

$$\mathbf{l} + \mathbf{v} = \mathbf{A} \cdot \hat{\mathbf{y}}. \tag{2.36}$$

Die Lösung für $\mathscr{L}_2$ Ausgleichungsprobleme (also die Lösung mit minimaler Verbesserungsquadratsumme) ist bekanntermaßen nach dem Gauss-Markov Modell (GMM) (Niemeier, 2008):

$$\hat{\mathbf{y}} = \left(\mathbf{A}^T \cdot \mathbf{P} \cdot \mathbf{A}\right)^{-1} \cdot \mathbf{A}^T \cdot \mathbf{P} \cdot \mathbf{l}. \tag{2.37}$$

Probleme, die nicht in der Form von Gleichung 2.36 vorliegen (vgl. Gauss-Helmert Modell, GHM), lassen sich in ein Gauss-Markov Modell zurückführen (Jäger et al., 2005).

Ist die Funktion f hingegen nichtlinear, so erfolgt eine Linearisierung von Gl. 2.31 mit den genäherten Parametern $\mathbf{y}_0$:

$$(\mathbf{l} - f(\mathbf{y}_0)) + \mathbf{v} = \mathrm{d}\mathbf{l} + \mathbf{v} = \mathbf{A} \cdot \mathrm{d}\hat{\mathbf{y}}. \tag{2.38}$$

Die JACOBI Matrix $\mathbf{A}$ besteht aus den Koeffizienten a_{ij} in Zeile i und Spalte j:

$$a_{ij} = \left(\frac{\partial f_i}{\partial y_j} \right)_{\mathbf{y}_0} . \tag{2.39}$$

Somit kann Gl. 2.37 für den nichtlinearen Fall genutzt werden:

$$\mathrm{d}\hat{\mathbf{y}} = \left(\mathbf{A}^T \cdot \mathbf{P} \cdot \mathbf{A} \right)^{-1} \cdot \mathbf{A}^T \cdot \mathbf{P} \cdot \mathrm{d}\mathbf{l}. \tag{2.40}$$

Über Gl. 2.40 wird $\mathrm{d}\hat{\mathbf{y}}$ bestimmt und auf die Näherungslösung addiert:

$$\hat{\mathbf{y}} = \mathbf{y}_0 + \mathrm{d}\hat{\mathbf{y}}. \tag{2.41}$$

Die Schritte 2.38, 2.39, 2.40 u. 2.41 werden wiederholt, bis die Zuschläge gegen Null gehen bzw. die gewünschte Genauigkeit erreicht ist.

Die Beschreibung der Verteilung in Form einer Kofaktormatrix $\mathbf{Q}_{yy}$ des geschätzten Vektors $\hat{\mathbf{y}}$ wird wie folgt berechnet:

$$\mathbf{Q}_{yy} = \left(\mathbf{A}^T \cdot \mathbf{P} \cdot \mathbf{A} \right)^{-1} . \tag{2.42}$$

Dies gilt nur für lineare Zusammenhänge. Im nichtlinearen Fall ist Gl. 2.42 nur eine Näherung der 1. Ordnung.

In der Praxis sollte auf die direkte Berechnung der Inversen von $\mathbf{A}^T \cdot \mathbf{A}$ verzichtet werden (Moler, 2004), da das Produkt der Designmatrix mit sich selbst zu kleinen Zahlen führen kann, die mit endlicher Präzision auf einem Rechner nicht mehr darstellbar sind[5]. Moler (2004) empfiehlt und beschreibt zur numerischen Berechnung von Gleichung 2.37 eine QR Zerlegung, nach Guennebaud und Jacob (2018) ist eine SVD Zerlegung allgemein die genaueste aber auch rechenaufwändigste Methode. Guennebaud und Jacob (2018) stimmen mit Moler (2004) dahingehend überein, dass die QR Methode einen guten Kompromiss zwischen Rechenaufwand und Genauigkeit darstellt. Die Methoden in Kapitel 5 erfordern eine strenge numerische Behandlung, in Abschnitt 5.1.8 wird auf das Thema weiter aufgebaut.

Es gibt eine Vielzahl weiterer Schätzmethoden, wie in Abschnitt 2.5 beschrieben, aber von besonderer Bedeutung für Kapitel 5 ist die $\mathscr{L}_1$ Norm, also die Minimierung der Residuenbetragssumme:

$$\sum_i |v_i| \rightarrow \mathrm{Min}. \tag{2.43}$$

[5] Das Maschinen Epsilon für Floating Point Zahlen in 32-Bit IEEE-754 Darstellung beträgt: 6×10^{-8}, für die 64-Bit Darstellung: 1×10^{-16} (Plato, 2000). Die numerischen Eigenschaften von Gleitkommazahlen sind im Detail jedoch komplex und lassen sich aufgrund der Exponentialdarstellung nicht auf einzelne Grenzwerte festlegen; ein guter Überblick zu dem Thema findet sich in Goldberg (1991). Grewal und Andrews (2001) schreiben: *„The effects of roundoff may be thought to be minor, but overlooking them could be a major blunder."*

bzw. der gewichteten Residuenbetragssumme:

$$\sum_i |\mathbf{P}^{\frac{1}{2}} \cdot \mathbf{v}|_i \rightarrow \text{Min.} \tag{2.44}$$

2.5 M-Schätzer und robuste M-Schätzer

Im Abschnitt 2.4 wird die $\mathscr{L}_1$ und $\mathscr{L}_2$ Norm hervorgehoben, diese können aber als Teil der Menge der sogenannten M-Schätzer verstanden werden. Huber (1964) benannte die M-Schätzer[6] in Anlehnung an die Maximum-Likelihood Schätzung von Fisher (1912). Die M-Schätzer bilden damit ein allgemeines algorithmisches Lösungskonzept für die Parameterschätzung (Jäger et al., 2005). Der Beitrag von Huber (1964) geht noch einen Schritt weiter: der Einfluss von beliebig großen Beobachtungsfehlern wird betrachtet und im Zuge dessen wird ein Fundament für die Klasse der *robusten* M-Schätzer geschaffen (Hampel et al., 1986). Robuste Schätzmethoden zeichnen sich dadurch aus, dass beliebig große Messfehler einen begrenzten Einfluss auf das Ergebnis haben. Hawkins (1980) definiert Beobachtungsfehler bzw. Outlier folgendermaßen:

> *An outlier is an observation which deviates so much from the other observations as to arouse suspicions that it was generated by a different mechanism.* (D. M. Hawkins)

Bevor jedoch weiter auf die robusten Eigenschaften eingegangen wird, sollen zuerst die Grundlagen der M-Schätzer betrachtet werden. Um im Verlauf des Abschnitts auf ein einheitliches algorithmisches Lösungskonzept zu gelangen, wird von Beobachtungen $\bar{\mathbf{l}}$ mit gleicher Genauigkeit ausgegangen. Also Beobachtungen mit einer Einheitskovarianzmatrix $\mathbf{C}_{\bar{l}\bar{l}} = \mathbf{I}$. Da Beobachtungen diese Anforderung i. A. nicht erfüllen, muss eine *Homogenisierung* durchgeführt werden. Diese wird im Abschnitt 2.6 gesondert betrachtet.

Die Maximum-Likelihood Schätzung führt eine Likelihood-Funktion L ein. Diese ist definiert als Dichtefunktion in Abhängigkeit der gesuchten Parameter $\mathbf{y}$ sowie der Beobachtungen $\bar{\mathbf{l}}$ (Jäger et al., 2005, S. 100):

$$L\left(\mathbf{y}, \bar{\mathbf{l}}\right) =: p\left(-\bar{\mathbf{A}} \cdot \mathbf{y} + \bar{\mathbf{l}}\right). \tag{2.45}$$

Die wahrscheinlichste Lösung wird gefunden durch den Vektor $\mathbf{y}$, bei dem die Wahrscheinlichkeitsdichtefunktion das globale Maximum erreicht:

$$\mathbf{y}_M = \text{argmax}\left\{ L\left(\mathbf{y}, \bar{\mathbf{l}}\right) \right\} \tag{2.46}$$

[6] In Originalschreibweise: *(M)*-estimator für *generalized maximum likelihood*.

Bzw. falls eine a priori Schätzung für $\mathbf{y}$ vorliegt, kann die Maximum Likelihood Schätzung für $\mathbf{y}$ zum Zeitpunkt k als Sonderfall der a posteriori Bayes-Filterung verstanden werden, vgl. Gl. 2.11, (Jäger, 2018):

$$\hat{\mathbf{y}}_{k,M} = \text{argmax}\Big\{p(\mathbf{l}_k \mid \mathbf{y}_k) \cdot bel(\mathbf{y}_k)^{-}\Big\} \tag{2.47}$$

Dieses Maximum kann für unimodale Verteilungen (wie beispielsweise der Normalverteilung) bestimmt werden durch das Gleichsetzen der partiellen Ableitungen von Gleichung 2.45 zu Null:

$$\mathbf{y}_M =: \frac{\partial \mathrm{L}\left(\mathbf{y}, \bar{\mathbf{l}}\right)}{\partial y_i} = 0 \Big|_{\mathbf{y}_M}. \tag{2.48}$$

Huber verallgemeinert diesen Ansatz durch das Einführen der konvexen *Verlustfunktion* ρ anstelle der Likelihood-Funktion L. Ziel der M-Schätzung ist die Bestimmung des Minimums ausgehend von einer gewählten Verlustfunktion[7] ρ, die für das i-te Residuum definiert ist als:

$$\rho\left(\bar{v}_i\right) = \rho\left(\bar{\mathbf{a}}_i \cdot \mathbf{y} - \bar{l}_i\right). \tag{2.49}$$

Der Vektor $\bar{\mathbf{a}}_i$ ist die i-te Zeile der homogenisierten Designmatrix $\bar{\mathbf{A}}$. Die Ableitung von ρ sei die Funktion ψ, auch *Einflussfunktion* genannt:

$$\psi\left(\bar{v}_i\right) = \frac{d\rho\left(\bar{v}_i\right)}{d\,\bar{v}_i}. \tag{2.50}$$

Somit ist der Vektor $\mathbf{y}$ gesucht, der über alle Beobachtungen hinweg zu einer minimalen Summe der Verluste führt:

$$\mathbf{y}_M =: \sum \rho\left(\bar{v}_i\right) = \text{Min.} \Big|_{\mathbf{y}_M}. \tag{2.51}$$

Huber (1964) schafft die Grundlagen für eine allgemeine Theorie der robusten Schätzer, indem er für die Ableitung der Verlustfunktion fordert, dass diese beschränkt ist und unterhalb einer Konstanten c bleibt:

$$|\psi\left(\bar{v}_i\right)| < |c|. \tag{2.52}$$

Als Grundlage für Abschnitt 5.1.7 soll nun ein allgemeingültiges algorithmisches Lösungskonzept für M-Schätzer betrachtet werden. Es gibt eine Vielzahl an Beiträgen, die sich mit dem Thema beschäftigen: Fletcher et al. (1971), Dennis und Welsch (1978), Wolfe (1979) oder Dutter und Huber (1981). Jäger

[7] engl.: *loss function*

et al. (2005) beschreiben das Verfahren wie folgt: Für die Lösung von Gleichung 2.51 wird zuerst die Einflussfunktion ψ in die j-te Zeile des folgenden Gleichungssystems für u Unbekannte in $\mathbf{y}$ eingesetzt mit n Beobachtungen:

$$\frac{\partial}{\partial y_j}\left(\sum_{i=1}^{n}\rho\left(\bar{v}_i(\mathbf{y})\right)\right)=0\Big|_{\mathbf{y}_M}. \tag{2.53}$$

Daraus folgt:

$$\sum_{i=1}^{n}\frac{\partial \rho_i}{\partial \bar{v}_i}\cdot\frac{\partial \bar{v}_i}{\partial y}=0 \tag{2.54}$$

$$\sum_{i=1}^{n}\psi(\bar{v}_i)\cdot\frac{\partial \bar{v}_i}{\partial y_j}=0. \tag{2.55}$$

Dieses nichtlineare Gleichungssystem lässt sich durch einen iterativen Algorithmus lösen, der an einer normalen Gauss-Markov Ausgleichung angelehnt ist. Die Messungen werden in jedem k-ten Schritt neu gewichtet anhand der aktuellen Residuen $\bar{\mathbf{v}}_k$ und der Einflussfunktion ψ. Das Verfahren wird wiederholt, bis die gewünschte Genauigkeit erreicht wird oder eine Iterationsgrenze von k überschritten wird.

$$\mathbf{y}_M^{k+1}=\left(\bar{\mathbf{A}}^T\cdot\mathbf{W}^k\cdot\bar{\mathbf{A}}\right)^{-1}\cdot\bar{\mathbf{A}}^T\cdot\bar{\mathbf{l}} \tag{2.56}$$

Die Gewichtsmatrix $\mathbf{W}^k$ ist eine diagonal besetzte Gewichtsmatrix, die in jeder k-ten Iteration nach folgendem Schema neu berechnet wird:

$$\mathbf{W}^k=\begin{pmatrix} w_1^k & 0 & \cdots & 0\\ 0 & w_2^k & \ddots & \vdots\\ \vdots & \ddots & \ddots & 0\\ 0 & \cdots & 0 & w_n^k\end{pmatrix} \tag{2.57}$$

mit der *Gewichtsfunktion*

$$w_i^k=\frac{\psi\left(\bar{v}_i^k\right)}{\bar{v}_i^k}. \tag{2.58}$$

Der allgemeingültige lineare Zusammenhang in Gleichung 2.56 bildet die Grundlage für die Fehlerfortpflanzung in Abschnitt 5.1.8. Für nichtlineare Ausgangsprobleme muss die Lösung über eine Kaskade errechnet werden. Eine äußere Schleife bildet die Jacobi Designmatrix des nichtlinearen funktionalen Zusammenhangs, während die innere Schleife das beschriebene Verfahren durchführt. Als Grundlage für Kapitel 5 sind im Folgenden die wichtigsten Eigenschaften der $\mathscr{L}_2$ und $\mathscr{L}_1$ M-Schätzer aufgelistet.

Die Methode der kleinsten Quadrate ($\mathscr{L}_2$ Norm) ist definiert über die Verlustfunktion

$$\rho(\bar{v}) = \frac{1}{2}\bar{v}^2 \tag{2.59}$$

mit der nicht begrenzten Einflussfunktion:

$$\psi(\bar{v}) = \bar{v} \tag{2.60}$$
$$w(\bar{v}) = 1. \tag{2.61}$$

Die $\mathscr{L}_2$ Norm ist folglich nicht robust.

Die $\mathscr{L}_1$ Norm ist definiert über die Verlustfunktion

$$\rho(\bar{v}) = |\bar{v}| \tag{2.62}$$

mit der an der Stelle 0 nicht definierten Einflussfunktion

$$\psi(\bar{v}) = \frac{\bar{v}}{|\bar{v}|} \tag{2.63}$$
$$w(\bar{v}) = \frac{1}{|\bar{v}|}. \tag{2.64}$$

Auf den $\mathscr{L}_1$ M-Schätzer wird in Kapitel 5.1 weiter aufgebaut.

Allgemein muss für einen Schätzer abgewogen werden zwischen höchstmöglicher Effizienz und Robustheit. Ein Mittelweg bietet der von Huber (1964) vorgestellte Huber-Schätzer mit einer zusammengesetzten Verlustfunktion:

$$\rho(\bar{v}) = \begin{cases} \frac{1}{2}\bar{v}^2 & |\bar{v}| \leq k \\ k|\bar{v}| - \frac{1}{2}k^2 & |\bar{v}| > k \end{cases} \tag{2.65}$$

und der Einflussfunktion:

$$\psi(\bar{v}) = \begin{cases} \bar{v} & |\bar{v}| \leq k \\ k \cdot \mathrm{sgn}(\bar{v}) & |\bar{v}| > k \end{cases}. \tag{2.66}$$

2.6 Homogenisierung / Dekorrelation

In Abschnitt 2.5 wird von Beobachtungen mit einer Einheitskovarianzmatrix $\mathbf{C}_{\bar{l}\bar{l}} = \mathbf{I}$ ausgegangen. Da dies oft nicht gegeben ist, muss zuerst eine Homogenisierung durchgeführt werden. Dieses Verfahren

verändert die Messungen sowie die Designmatrix über eine lineare Transformation derartig, dass die Komponenten des Beobachtungsvektors voneinander dekorreliert und gleichgenau sind.

Wicki (1998) geht dabei von einer rein diagonal besetzten Kovarianzmatrix aus und zieht entlang der Inversen dieser Matrix die Wurzel um somit auf einfache Art die Homogenisierung durchzuführen:

$$\mathbf{P} = \mathbf{C}_{ll}^{-1} \tag{2.67}$$

$$\mathbf{T} = \operatorname{diag}\left(\sqrt{p_1}, \sqrt{p_2}, \ldots, \sqrt{p_n}\right). \tag{2.68}$$

Die Homogenisierung erfolgt dann als:

$$\bar{\mathbf{l}} = \mathbf{T} \cdot \mathbf{l} \tag{2.69}$$

$$\bar{\mathbf{A}} = \mathbf{T} \cdot \mathbf{A}. \tag{2.70}$$

Nicht zu vergessen ist, dass nach der Homogenisierung sich auch die Residuen ändern, beziehungsweise gesondert zu behandeln sind:

$$\bar{\mathbf{v}} = \mathbf{T} \cdot \mathbf{v}. \tag{2.71}$$

Der Ansatz von Wicki (1998) ist für den genannten Spezialfall der rein diagonal besetzten Kovarianzmatrix durchaus empfehlenswert, aber nicht allgemeingültig. Jäger et al. (2005) schlagen ein Verfahren auf Basis einer diagonalen Eigenwertmatrix ($\mathbf{\Lambda}$) und Modalmatrix ($\mathbf{M}$) Faktorisierung vor:

$$\mathbf{C}_{ll} = \mathbf{M} \cdot \mathbf{\Lambda} \cdot \mathbf{M}^T \tag{2.72}$$

$$\mathbf{\Lambda}^{-\frac{1}{2}} = \operatorname{diag}\left(\sqrt{\Lambda_1}, \sqrt{\Lambda_2}, \ldots, \sqrt{\Lambda_n}\right) \tag{2.73}$$

$$\mathbf{C}_{ll} = \mathbf{M} \cdot \mathbf{\Lambda}^{-\frac{1}{2}} \cdot \mathbf{M}^T. \tag{2.74}$$

Dieses Verfahren ist auch bei vollbesetzten Kovarianzmatrizen allgemeingültig. Im Rahmen dieser Arbeit wurde eine zu Jäger et al. (2005) äquivalente und allgemeingültige Homogenisierung auf Basis der Cholesky Zerlegung verwendet. Die Cholesky Zerlegung setzt eine symmetrische positiv definite Matrix voraus, was bei Kovarianzmatrizen gegeben ist. Somit kann folgende Zerlegung durchgeführt werden:

$$\mathbf{Q}_{ll} = \mathbf{L} \cdot \mathbf{L}^{\mathrm{T}} \tag{2.75}$$

$$\mathbf{P}_{ll} = \mathbf{Q}_{ll}^{-1} \tag{2.76}$$

$$\mathbf{P}_{ll} = \left(\mathbf{L}^T\right)^{-1} \cdot \mathbf{L}^{-1} \tag{2.77}$$

$$\mathbf{L}_w = \mathbf{L}^{-1}. \tag{2.78}$$

Die Homogenisierung erfolgt analog zu Gleichung 2.69 und 2.70:

$$\bar{\mathbf{l}} = \mathbf{L}_w \cdot \mathbf{l} \tag{2.79}$$

$$\bar{\mathbf{A}} = \mathbf{L}_w \cdot \mathbf{A}. \tag{2.80}$$

Die Inverse in Gl. 2.78 muss dabei nicht explizit gebildet werden, sondern kann gleich als lineares Gleichungssystems verstanden werden, bei dem $\bar{\mathbf{A}}$ und $\bar{\mathbf{l}}$ gesucht sind[8].

Verwandt mit der Homogenisierung ist die reine Dekorrelation von Messungen, wobei hier tatsächlich nur die Dekorrelation durchgeführt wird, d. h. die Messungen haben hier anschließend nicht alle die gleiche Genauigkeit. Grewal und Andrews (2001) führen dazu eine $\mathbf{UDU}^T$ Zerlegung durch:

$$\mathbf{Q}_{ll} = \mathbf{U} \cdot \mathbf{D} \cdot \mathbf{U}^T \tag{2.81}$$

$$\bar{\mathbf{l}} = \mathbf{U}^{-1} \cdot \mathbf{l} \tag{2.82}$$

$$\bar{\mathbf{A}} = \mathbf{U}^{-1} \cdot \mathbf{A}. \tag{2.83}$$

Dies kann genutzt werden für die sequentielle Verarbeitung von Messwerten im Rahmen einer Ausgleichung nach vermittelnden Beobachtungen oder einer a posteriori Kalman-Filter Zustandsschätzung. Grewal und Andrews (2001) empfehlen dies ausdrücklich, da die Gesamtzahl der mathematischen Operationen trotz der Dekorrelation insgesamt sinkt.

2.7 Integer Least-Squares

Bisher wurde für die Schätzprobleme von reellen Zahlen ausgegangen ($\mathbb{R}$), aber im Hinblick auf den Simplex-Algorithmus in Kapitel 5 (der dem Bereich *Linear Programming* angehört) wird in diesem Abschnitt auf die Möglichkeit eingegangen, dass mit der Methode der kleinsten Quadrate die Lösungsmenge auf ganze Zahlen ($\mathbb{Z}$) eingeschränkt werden kann, basierend auf den Grundlagen und Ideen von Teunissen (2004).

Gesucht sei ein Unbekanntenvektor $\hat{\mathbf{x}}$, der zum Teil aus ganzen Zahlen besteht $\hat{\mathbf{a}}$ (Integer) und zum Teil aus reellen Zahlen $\hat{\mathbf{b}}$:

$$\hat{\mathbf{x}} = \begin{pmatrix} \hat{\mathbf{a}} \\ \hat{\mathbf{b}} \end{pmatrix}, \hat{\mathbf{a}} \in \mathbb{Z}, \hat{\mathbf{b}} \in \mathbb{R}. \tag{2.84}$$

Das funktionale Modell ist durch die $\mathbf{A}$ Matrix gegeben. Diese Matrix beschreibt sowohl die reellen, als auch die ganzzahligen Zusammenhänge. Mit dem bekannten Modell der vermittelnden Ausgleichung:

$$\mathbf{l} + \mathbf{v} = \mathbf{A} \cdot \hat{\mathbf{x}}. \tag{2.85}$$

[8] Siehe Quellcode in Anhang A.

Das Problem wird zuerst als normales Ausgleichungsproblem gelöst. Die Ergebniskovarianz ist folgendermaßen segmentiert:

$$\mathbf{Q}_{yy} = \left(\mathbf{A}^T \mathbf{Q}_{ll}^{-1} \mathbf{A}\right)^{-1} = \begin{pmatrix} \mathbf{Q}_{aa} & \mathbf{Q}_{ab} \\ \mathbf{Q}_{ba} & \mathbf{Q}_{bb} \end{pmatrix}. \tag{2.86}$$

Der Lösungsvektor $\hat{\mathbf{a}}$ ist jetzt noch nicht auf ganze Zahlen festgelegt, deshalb wird im nächsten Schritt ein Vektor $\bar{\mathbf{a}}_i$ gesucht, der nur aus ganzen Zahlen besteht und möglichst geringe Residuen $\mathbf{v}_i$ erzeugt.

$$\mathbf{v}_i = (\hat{\mathbf{a}} - \bar{\mathbf{a}}_i)^T \mathbf{Q}_{aa}^{-1} (\hat{\mathbf{a}} - \bar{\mathbf{a}}_i) \tag{2.87}$$

$$\mathbf{v}_i \rightarrow \text{Min.} \tag{2.88}$$

Der Vektor $\bar{\mathbf{a}}_i$ mit den geringsten Residuen im Suchraum ist dann der gesuchte ganzzahlige Vektor $\bar{\mathbf{a}}$. Der reelle Teil $\hat{\mathbf{b}}$ muss entsprechend verbessert werden:

$$\bar{\mathbf{b}} = \hat{\mathbf{b}} - \mathbf{Q}_{ba} \mathbf{Q}_a^{-1} (\hat{\mathbf{a}} - \bar{\mathbf{a}}). \tag{2.89}$$

Speziell für GNSS Auswertungen, bei denen die Phasenmehrdeutigkeiten gelöst werden müssen, ist ILS von großer Bedeutung. Um mit möglichst geringem Rechenaufwand eine Lösung für 2.88 zu finden, eignet sich z. B. die MLAMBA Methode (Chang et al., 2005). Wie eingangs erwähnt, kann der Bereich *Integer Programming* (IP) dem *Linear Programming* (LP) zugeordnet werden. Im Hinblick auf die vorgestellten Methoden der *linearen Optimierung* in Kapitel 5 sei an dieser Stelle darauf hingewiesen, dass die in dieser Arbeit vorgestellten Methoden um Ganzzahlbedingungen erweitert werden können. Williams (2009) beschreibt LP Probleme als *einfach*, IP Probleme hingegen als *schwierig* und schreibt „*For IP [...] no algorithms have ever been found which are not exponential or worse.*“ (Williams, 2009, Kap. 2.4.2). Xu et al. (2010) zeigen, wie sich IP Methoden auf das Finden von GNSS Trägerphasenmehrdeutigkeiten grundsätzlich anwenden lassen.

2.8 Kalman-Filter

Bei vorliegender Gauß-Verteilung ist das Standardwerkzeug zur Zustandsschätzung von dynamischen Systemen das Kalman-Filter (Kálmán, 1960), benannt nach Rudolf Emil Kálmán[9]. Kurz nach der Veröffentlichung besuchte Kálmán den leitenden Angestellten Stanley F. Schmidt am NASA AMES Research Center. Schmidt nannte es rückblickend einen erstaunlichen Zufall (McGee und Schmidt, 1985), dass Kálmán mit seiner Veröffentlichung genau die Probleme behandelte, die den NASA Forschern damals Schwierigkeiten bereitete. Insbesondere die sequentielle Verarbeitung war attraktiv, da die Computer wenig Speicher und Rechenleistung zur Verfügung hatten. Schmidt setzte seine wichtigsten Mitarbeiter auf Kálmáns Beitrag an, die allerdings Mühe hatten die Veröffentlichung zu verstehen und umzusetzen (McGee und Schmidt, 1985). Grund dafür war die ungewöhnliche Notation, sowie das damals neue

[9] *1930 Budapest, †2016 Florida.

Konzept der Zustandsraumdarstellung. Die Forscher und Ingenieure waren letztendlich aber erfolgreich und konnten ihre Ergebnisse u. a. im Apollo Navigationssystem einsetzen[10]. Seit der Apollo Mission ist das Kalman-Filter aus der Navigationszustandsschätzung nicht mehr wegzudenken. Das Kalman-Filter ist optimal für die Zustandsschätzung von linearen dynamischen Systemen, die nur durch weißes normalverteiltes Rauschen beeinflusst werden (Wendel, 2011). D. h. unter diesen Einschränkungen gibt es keinen Schätzer, der genauere Ergebnisse liefert, vgl. Abschnitt 2.4.

Grewal und Andrews (2001) sehen daher das Kalman-Filter als eine der wichtigsten Entdeckungen des 20. Jahrhunderts an. Die folgenden Abschnitte betrachten die für diese Arbeit relevanten Aspekte bei der Navigationszustandsschätzung. Kalman-Filter werden daher in der Literatur sehr tiefgehend untersucht, z. B. in Gelb (1974), Maybeck (1979), Levy (1997), Grewal und Andrews (2001) oder Farrell (2008).

Im Kern findet in einem Kalman-Filter eine Ausgleichung nach der Methode der kleinsten Quadrate statt und das Kalman-Filter kann als Sonderfall einer vermittelnden Ausgleichung angesehen werden. Dies wird z. B. in Hofmann-Wellenhof et al. (2003) für die theoretische Herleitung des Kalman-Filters genutzt. Diese Herangehensweise stellt eine wichtige Grundlage für Kapitel 5 dar und wird daher näher beleuchtet. Andere Ansätze zur Herleitung des Kalman-Filters basieren auf dem Satz von Bayes (Metzger, 2006), einem Orthogonalitätsprinzip[11] bei der Originalveröffentlichung (Kálmán, 1960) oder über die Fehlerfortpflanzung von normalverteilten Zufallsvektoren (Wendel, 2011).

Für eine a priori Zustandsschätzung $\mathbf{y}^-$ zum aktuellen Zeitpunkt k liegt ein Satz von Beobachtungen in Form des Messvektors $\mathbf{l}$ vor. Dabei wird angenommen, dass die Zusammenhänge linear sind. Diese Beobachtungen sollen nun die a priori Zustandsschätzung verbessern. Der Übersichtlichkeit halber wird in diesem Abschnitt angenommen, dass sich die Zusammenhänge auf denselben Zeitpunkt bzw. dieselbe Epoche k beziehen, wenn kein Index k die Epoche kennzeichnet. Die a posteriori Zustandsschätzung $\mathbf{y}^+$ lässt sich mit dem Gauss-Markov Modell (GMM) berechnen, indem die a priori Schätzung als Beobachtung selbst in die Ausgleichung einfließt:

$$\underbrace{\begin{bmatrix} \mathbf{y}_k^- \\ \mathbf{l}_k \end{bmatrix}}_{\mathbf{l}_\star} = \underbrace{\begin{bmatrix} \mathbf{I} \\ \mathbf{A}_k \end{bmatrix}}_{\mathbf{A}_\star} \cdot \mathbf{y}_k^+ . \tag{2.90}$$

Liegen Beobachtungen vor, die über eine Funktion h in einem nichtlinearen Zusammenhang zu dem Systemzustand stehen:

$$\mathbf{l}_k + \mathbf{v}_k = h(\mathbf{y}_k), \tag{2.91}$$

[10] Im Zuge dessen wurde die erste Square-Root Kalman-Filter Formulierung entwickelt. Auch wurden robuste Eigenschaften in das Kalman-Filter der Apollo Mission eingebaut: Messungen wurden verworfen, wenn die Residuen einen 3-σ Test nicht bestanden haben (McGee und Schmidt, 1985).

[11] Siehe (Grewal und Andrews, 2001, Kap. 3.9.2).

so kann die Matrix $\mathbf{A}$ über eine JACOBI-Matrix angenähert werden:

$$\mathbf{A}_k = \left.\frac{\partial h(\mathbf{y}_k)}{\partial \mathbf{y}}\right|_{\mathbf{y}=\mathbf{y}_k^-}. \tag{2.92}$$

Gleichung 2.90 kann dann zu einem linearisierten Modell erweitert werden, vergleichbar mit dem allgemeinen Aufbau von Gleichung 2.36:

$$\begin{bmatrix}\mathbf{y}_k^- \\ \mathbf{l}_k\end{bmatrix} + \begin{bmatrix}\mathbf{v}_{y_k^-} \\ \mathbf{v}_{l_k}\end{bmatrix} = \begin{bmatrix}\mathbf{I} \\ \mathbf{A}_k\end{bmatrix} \cdot \delta\mathbf{y}_k^+ + \begin{bmatrix}\mathbf{y}_k^- \\ \mathbf{l}\left(\mathbf{y}_k^-\right)\end{bmatrix}. \tag{2.93}$$

Der Korrekturschritt zum Zeitpunkt k kann als Bestimmung des Korrekturvektors $\delta\mathbf{y}^+$ angesehen werden:

$$\mathbf{y}_k^+ = \mathbf{y}_k^- + \delta\mathbf{y}_k^+. \tag{2.94}$$

Mit der Differenz $\Delta\mathbf{l}_k$ zwischen Beobachtung und Vorhersage kann im Folgenden ein Ausgleichungsproblem formuliert werden.

$$\Delta\mathbf{l}_k = \mathbf{l}_k - \mathbf{l}(\mathbf{y}_k^-). \tag{2.95}$$

$$\delta\mathbf{y}_k^+ = \underbrace{\left[\begin{pmatrix}\mathbf{I}\\ \mathbf{A}_k\end{pmatrix}^T \cdot \begin{pmatrix}\mathbf{Q}_{yy,k}^- & \mathbf{0}\\ \mathbf{0} & \mathbf{Q}_{ll,k}\end{pmatrix}^{-1} \cdot \begin{pmatrix}\mathbf{I}\\ \mathbf{A}_k\end{pmatrix}\right]^{-1}}_{=\left[\left(\mathbf{Q}_{yy,k}^-\right)^{-1}+\mathbf{A}_k^T\cdot\mathbf{Q}_{ll,k}^{-1}\cdot\mathbf{A}_k\right]^{-1}=\mathbf{Q}_{yy,k}^+} \cdot \underbrace{\begin{pmatrix}\mathbf{I}\\ \mathbf{A}_k\end{pmatrix}^T \cdot \begin{pmatrix}\mathbf{Q}_{yy,k}^- & \mathbf{0}\\ \mathbf{0} & \mathbf{Q}_{ll,k}\end{pmatrix}^{-1} \cdot \begin{pmatrix}\mathbf{0}\\ \Delta\mathbf{l}_k\end{pmatrix}}_{=\mathbf{A}^T\cdot\mathbf{Q}_{ll,k}^{-1}\cdot\Delta\mathbf{l}_k} \tag{2.96}$$

Darauf aufbauend lässt sich der a posteriori Zustandsvektor $\mathbf{y}^+$ zur Epoche k vereinfacht berechnen:

$$\mathbf{y}_k^+ = \mathbf{y}_k^- + \underbrace{\mathbf{Q}_{yy,k}^+ \cdot \mathbf{A}_k^T \cdot \mathbf{Q}_{ll,k}^{-1}}_{\mathbf{K}_1} \cdot \Delta\mathbf{l}_k \tag{2.97}$$

Die Woodbury-Matrix-Identität (Woodbury, 1950) stellt folgende allgemeine Gleichung für die beliebigen Matrizen $\mathbf{D}, \mathbf{E}, \mathbf{F}$ und $\mathbf{G}$ auf (mit entsprechend passenden Dimensionen):

$$\left(\mathbf{E}^{-1} + \mathbf{F}\cdot\mathbf{H}^{-1}\cdot\mathbf{G}\right)^{-1} = \mathbf{E} - \mathbf{E}\cdot\mathbf{F}\cdot(\mathbf{H}+\mathbf{G}\cdot\mathbf{E}\cdot\mathbf{F})^{-1}\cdot\mathbf{G}\cdot\mathbf{E}. \tag{2.98}$$

Darauf aufbauend lässt sich zeigen, dass sich $\mathbf{K}_1$ in die häufig genutzte Darstellung $\mathbf{K}_k$ umformen lässt:

$$\mathbf{K}_1 = \mathbf{K}_k = \left[\left(\mathbf{Q}_{yy,k}^-\right)^{-1} + \mathbf{A}_k^T\cdot\mathbf{Q}_{ll,k}^{-1}\cdot\mathbf{A}_k\right]^{-1}\cdot\mathbf{A}_k^T\cdot\mathbf{Q}_{ll,k}^{-1} = \mathbf{Q}_{yy,k}^-\cdot\mathbf{A}_k^T\cdot\left(\mathbf{Q}_{ll,k}+\mathbf{A}_k\cdot\mathbf{Q}_{yy,k}^-\cdot\mathbf{A}_k^T\right)^{-1}. \tag{2.99}$$

$\mathbf{K}_1$ entspricht der Darstellung von (Gelb, 1974, Kap. 4) oder Jäger (2018).

Die a posteriori Kovarianzmatrix des Zustandsvektors $\mathbf{Q}_{yy}^{+}$ kann zudem ebenfalls auf eine weitere Art dargestellt werden, mit demselben Ergebnis.

$$\mathbf{Q}_{yy,k}^{+} = (\mathbf{I} - \mathbf{K}_k \cdot \mathbf{A}_k) \cdot \mathbf{Q}_{yy,k}^{-} \tag{2.100}$$

dies wird ausmultipliziert zu:

$$\mathbf{Q}_{yy,k}^{+} = \mathbf{Q}_{yy,k}^{-} - \underbrace{\mathbf{Q}_{yy,k}^{-} \cdot \mathbf{A}_k^T \left(\mathbf{A}_k \cdot \mathbf{Q}_{yy,k}^{-} \cdot \mathbf{A}_k^T + \mathbf{Q}_{ll,k}\right)^{-1}}_{\mathbf{K}_k} \cdot \mathbf{A}_k \cdot \mathbf{Q}_{yy,k}^{-}. \tag{2.101}$$

Und anschließend, durch Ausnutzung der Woodbury-Matrix-Identität (siehe Gleichung 2.98), umgeformt zu:

$$\mathbf{Q}_{yy,k}^{+} = \left[\left(\mathbf{Q}_{yy,k}^{-}\right)^{-1} + \mathbf{A}_k^T \cdot \mathbf{Q}_{ll,k}^{-1} \cdot \mathbf{A}_k\right]^{-1}. \tag{2.102}$$

Dies ergibt nun einen Korrekturschritt für die Epoche k, auch Estimationsschritt oder Fusionsschritt genannt:

$$\mathbf{y}_k^{+} = \mathbf{y}_k^{-} + \mathbf{K}_k \cdot \left(\mathbf{l}_k - h(\mathbf{y}_k^{-})\right). \tag{2.103}$$

Durch diesen Korrekturschritt verbessert sich die Genauigkeit der a posteriori Schätzung, das ist an der Form von Gleichung 2.100 zu sehen. Abbildung 2.5 zeigt diesen Sachverhalt anhand der Wahrscheinlichkeitsdichtefunktion. Der geschätzte a priori Zustand $\mathbf{y}^{-}$ wird mit der Messung $\mathbf{l}$ und Gl. 2.103 zu einem a posteriori Zustand $\mathbf{y}^{+}$ fusioniert. Die Kombination von Gleichung 2.100 und 2.103 entspricht dem Korrekturschritt der Bayes Filterung (vgl. Gl. 2.11):

$$bel(\mathbf{y}_k)^{+} = \eta \cdot p(\mathbf{l}_k \mid \mathbf{y}_k) \cdot bel(\mathbf{y}_k)^{-}. \tag{2.104}$$

Der Korrekturschritt allein macht aber noch nicht das Kalman-Filter aus. Die Modellierung der Systemdynamik und die damit einhergehende Vorhersage (a priori Schätzung) des Zustandsvektors und dessen Kovarianzmatrix unterscheiden das Kalman-Filter von einer rekursiven Ausgleichung nach der Methode der kleinsten Quadrate. Für eine Vorhersage muss eine geeignete Modellierung des zeitlichen Verhaltens für den Systemzustand $\mathbf{y}$ vorliegen. Essentiell wichtig ist die passende stochastische Beschreibung der Verteilung in Form der Kovarianzmatrix. Das setzt ganz allgemein voraus, dass sich die Verteilung über eine Kovarianzmatrix beschreiben lässt. Für lineare Zusammenhänge ist das mit Hilfe der Fehlerfortpflanzung optimal möglich:

$$\mathbf{y}_k^{-} = \mathbf{\Phi}_{k-1} \cdot \mathbf{y}_{k-1}^{+} + \mathbf{B}_{k-1} \cdot \mathbf{u}_{k-1} + \mathbf{G}_{k-1} \cdot \mathbf{w}_{k-1} \tag{2.105}$$

$$\mathbf{Q}_{yy,k}^{-} = \mathbf{\Phi}_{k-1} \cdot \mathbf{Q}_{yy,k-1}^{+} \cdot \mathbf{\Phi}_{k-1}^T + \mathbf{G}_{k-1} \cdot \mathbf{Q}_{k-1} \cdot \mathbf{G}_{k-1}^T. \tag{2.106}$$

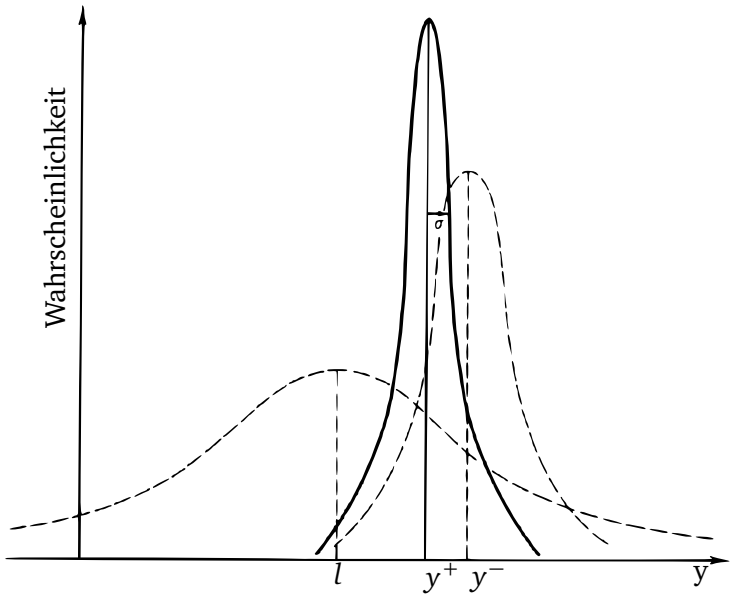


Abbildung 2.5.: A priori und a posteriori Wahrscheinlichkeiten durch den Korrekturschritt, nach einer Abbildung von Maybeck (1979).

Die Matrix $\mathbf{Q}_k$ beschreibt einen nicht näher spezifizierten diskreten Rauschanteil im Vektor $\mathbf{w}_k$. Dieser Rauschanteil beschreibt Unsicherheiten in der Vorhersage. Beispielsweise wenn die Systemdynamik nur näherungsweise bekannt ist. Die Matrix $\mathbf{G}_k$ modelliert den Zusammenhang zwischen $\mathbf{w}_k$ und dem Zustandsvektor. Unsicherheiten in der Fehlerverteilung können hier (zu einem gewissen Grad) kompensiert werden. Diese (nicht strenge) Modellierung von $\mathbf{Q}_k$ wird auch als Filter-Tuning bezeichnet (Wendel, 2011). Der optionale Vektor mit Eingangsgrößen $\mathbf{u}_k$ (engl.: *system input vector*) enthält externe Größen, die zu der Vorhersage beitragen. Die sogenannte Eingangsmatrix $\mathbf{B}_k$ modelliert diese Beiträge. Abbildung 2.6 zeigt, wie sich die Unsicherheiten in $\mathbf{y}_k$ mit jedem Vorhersageschritt (Epoche k von 0 bis 2) erhöhen. Gleichung 2.105 u. 2.106 können als Sonderfall von Gl. 2.12 verstanden werden und lösen die

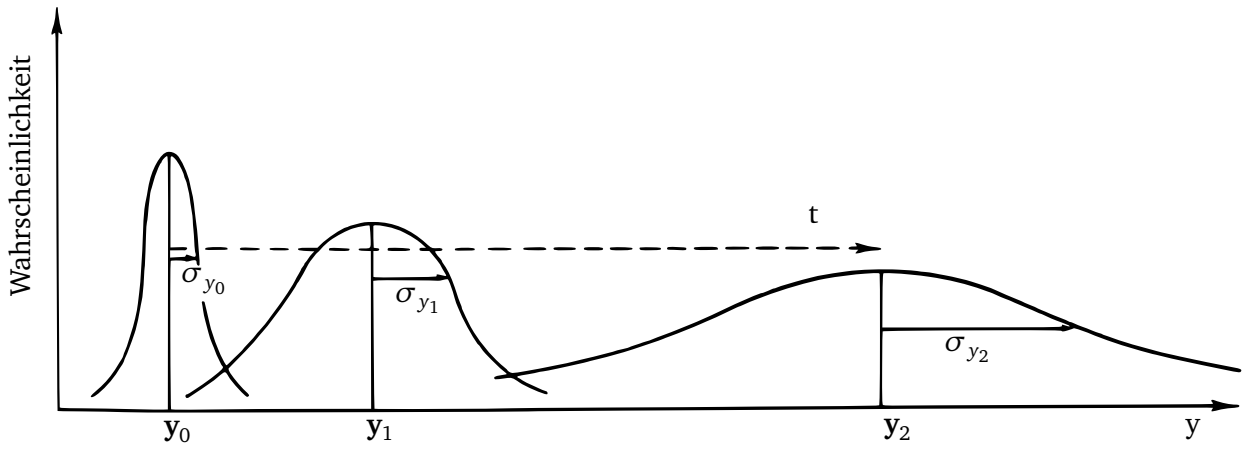


Abbildung 2.6.: Kalman-Filter Vorhersageschritt, nach einer Abbildung von Maybeck (1979).

Chapman-Kolmogorov Gleichung für lineare oder linearisierte Systeme mit Normalverteilung:

$$bel(\mathbf{y}_k)^- = \int p(\mathbf{y}_k \mid \mathbf{y}_{k-1}, \mathbf{u}_{k-1}) \cdot bel(\mathbf{y}_{k-1})^+ \, \mathrm{d}\mathbf{y}_{k-1} \tag{2.107}$$

Im Ergebnis entsteht die wiederum normalverteilte Vorhersage $bel(\mathbf{y}_k)^-$, siehe Metzger (2006). Laut Grewal et al. (2007) sind für einfache Anwendungen die von Kálmán (1960) publizierten Gleichungen (2.103) numerisch stabil, wenn dabei die Symmetrie der Kovarianzmatrix sichergestellt wird. Entweder in dem nur die Dreiecksmatrix gespeichert wird oder durch ein Abgleichen nach jedem x-ten Filterschritt:

$$\mathbf{Q}_{yy} \leftarrow \frac{\mathbf{Q}_{yy} + \mathbf{Q}_{yy}^T}{2}. \tag{2.108}$$

In Fällen, in denen die Matrizen schlecht konditioniert sind, d. h. die numerischen Unterschiede liegen nahe an der Fließkomma-Rechengenauigkeit oder haben große Unterschiede in den Größenordnungen, kann eine stabilisierte Implementierung bzw. eine abweichende Berechnungsvorschrift von Vorteil sein. In Grewal et al. (2007) findet sich eine Übersicht und ein Vergleich verschiedener Implementierungen:

- Konventionelle Kalman-Filter Gleichungen, publiziert von Kálmán (1960)
- Swerling Inverse Form (vor Kalman publiziert von P. Swerling)
- Joseph's Form (von P. D. Joseph)
- Joseph-Bierman (modifiziert von G. J. Bierman)
- Joseph-De Vries (modifiziert von T. W. DeVries)
- Potter (J. E. Potter Cholesky-basiert, verwendet in den Apollo Mond Missionen)
- Carlson („triangular-method" von N. A. Carlson)
- Bierman (**U-D** Algorithmus von G. J. Biermann)

Die Methoden von Carlson und Bierman sind in dem Test von Grewal et al. (2007) numerisch am stabilsten. Die Autoren betonen aber, dass dies keine allgemeine Empfehlung für alle Anwendungsfälle darstellt.

Eine vergleichsweise neue Darstellung wurde von Takasu (2012) vorgestellt. Diese Darstellung von Takasu (2012) wurde für Vergleiche mit einem $\mathscr{L}_2$ Kalman-Filter in dieser Arbeit gewählt. Der Grund ist, dass diese Form numerisch sehr stabil ist und sich gleichzeitig sehr gut mit den optimierten Mathematikbibliotheken LAPACK/BLAS (Anderson et al., 1999) oder EIGEN (Guennebaud und Jacob, 2018) umsetzen lässt. Bei dem Verfahren wird nur mit der oberen Dreiecksmatrix der Kovarianzmatrix gerechnet, wodurch Rechenzeit eingespart wird. Der Schritt in Gl. 2.108 muss nicht durchgeführt werden, die Symmetrie bleibt automatisch erhalten.

Daher soll diese Form im Detail dargestellt werden. In der rechten Spalte ist die passende Funktion der LAPACK/BLAS Bibliothek angegeben:

$$\mathbf{D} = \mathbf{Q}_{yy}^{-} \cdot \mathbf{A}^T \qquad \texttt{symm()} \text{ - Symmetric Matrix Product} \tag{2.109a}$$

$$\mathbf{S} = \mathbf{A} \cdot \mathbf{D} + \mathbf{Q}_{ll} \qquad \texttt{gemm()} \text{ - General Matrix Product} \tag{2.109b}$$

$$\mathbf{U} = \mathit{cholesky}(\mathbf{S}) \qquad \texttt{potrf()} \text{ - Cholesky Factorization} \tag{2.109c}$$

$$\mathbf{E} = \mathbf{D} \cdot \mathbf{U}^{-1} \qquad \texttt{trsm()} \text{ - Solving Triangular Matrix} \tag{2.109d}$$

$$\mathbf{K} = \mathbf{E} \cdot (\mathbf{U}^{-1})^T \qquad \texttt{trsm()} \text{ - Solving Triangular Matrix} \tag{2.109e}$$

$$\mathbf{y}^{+} = \mathbf{y}^{-} + \mathbf{K} \cdot \mathbf{l} \qquad \texttt{gemv()} \text{ - Matrix Vector Product} \tag{2.109f}$$

$$\mathbf{Q}_{yy}^{+} = \mathbf{Q}_{yy}^{-} - \mathbf{E} \cdot \mathbf{E}^T. \qquad \texttt{syrk()} \text{ - Symmetric Rank Update} \tag{2.109g}$$

Die Funktion *cholesky()* in Gleichung 2.109c steht für die Cholesky-Faktorisierung: $\mathbf{U}^T\mathbf{U} = \mathbf{S}$. Bei n Variablen im Zustandsvektor $\mathbf{y}$ und m Beobachtungen im Vektor $\mathbf{l}$ ergeben sich die Matrixdimensionen: $\mathbf{D} : n \times m$, $\mathbf{S} : m \times m$, $\mathbf{U} : m \times m$, $\mathbf{E} : n \times m$ und $\mathbf{K} : n \times m$.

> *The update form may be my original. I checked Gelb, Grewal, Gibbs, Simon, Crassidis and papers by Google. But I couldn't find any articles for such form in previous works. One motivation is numerical stability. In terms of the computation cost, the form may be the best. It also can easily use optimized matrix libraries by LAPACK/BLAS.*
>
> *(*E-Mail von Tomoji Takasu an den Autor am 26. August 2012*)*

Die Gleichungen 2.109a bis 2.109g lassen sich einfach verifizieren:

$$\mathbf{K} = \underbrace{\mathbf{Q}_{yy}^{-} \cdot \mathbf{A}^T}_{=\mathbf{D}} \left(\underbrace{\mathbf{A} \cdot \mathbf{Q}_{yy}^{-} \cdot \mathbf{A}^T + \mathbf{Q}_{ll}}_{=\mathbf{S}} \right)^{-1} \tag{2.110a}$$

$$\mathbf{S} = \mathbf{A} \cdot \mathbf{D} + \mathbf{Q}_{ll} \tag{2.110b}$$

$$\mathbf{S} = \mathbf{U}^T \cdot \mathbf{U} \tag{2.110c}$$

$$\mathbf{U} = \text{cholesky}(\mathbf{S}) \tag{2.110d}$$

$$\mathbf{U}^{-1} = \text{triangular-invert}(\mathbf{U}) \tag{2.110e}$$

$$\mathbf{S}^{-1} = \left(\mathbf{U}^T \cdot \mathbf{U}\right)^{-1} = \mathbf{U}^{-1} \cdot \left(\mathbf{U}^{-1}\right)^T \tag{2.110f}$$

$$\mathbf{E} = \mathbf{D} \cdot \mathbf{U}^{-1} \tag{2.110g}$$

$$\mathbf{K} = \mathbf{E} \cdot \left(\mathbf{U}^{-1}\right)^T. \tag{2.110h}$$

Ebenso die Modifikation der Kovarianzmatrix:

$$\mathbf{Q}_{yy}^{+} = (\mathbf{I} - \mathbf{K} \cdot \mathbf{A}) \cdot \mathbf{Q}_{yy}^{-} \tag{2.111a}$$

$$\mathbf{E} \cdot \mathbf{E}^{T} = \mathbf{D} \cdot \mathbf{U}^{-1} \cdot \left(\mathbf{D} \cdot \mathbf{U}^{-1}\right)^{T} = \mathbf{D} \cdot \mathbf{U}^{-1} \cdot \left(\mathbf{U}^{-1}\right)^{T} \cdot \mathbf{D}^{T} \tag{2.111b}$$

$$= \mathbf{D} \cdot \mathbf{S}^{-1} \cdot \mathbf{D}^{T} \tag{2.111c}$$

$$= \mathbf{K} \cdot \left(\mathbf{Q}_{yy}^{-} \cdot \mathbf{A}^{T}\right)^{T} \tag{2.111d}$$

$$= \mathbf{K} \cdot \mathbf{A} \cdot \mathbf{Q}_{yy}^{-}. \tag{2.111e}$$

Die a posteriori Kovarianzmatrix des Zustandsvektors lässt sich damit über ein *Symmetric Rank Update* durchführen:

$$\mathbf{Q}_{yy}^{+} = \mathbf{Q}_{yy}^{-} - \mathbf{E} \cdot \mathbf{E}^{T}. \tag{2.112}$$

Hier wird ebenfalls Rechenzeit eingespart, da das Produkt $\mathbf{E} \cdot \mathbf{E}^{T}$ nicht explizit berechnet werden muss, siehe Gl. 2.109g.

2.9 Nichtlineare Kalman-Filter

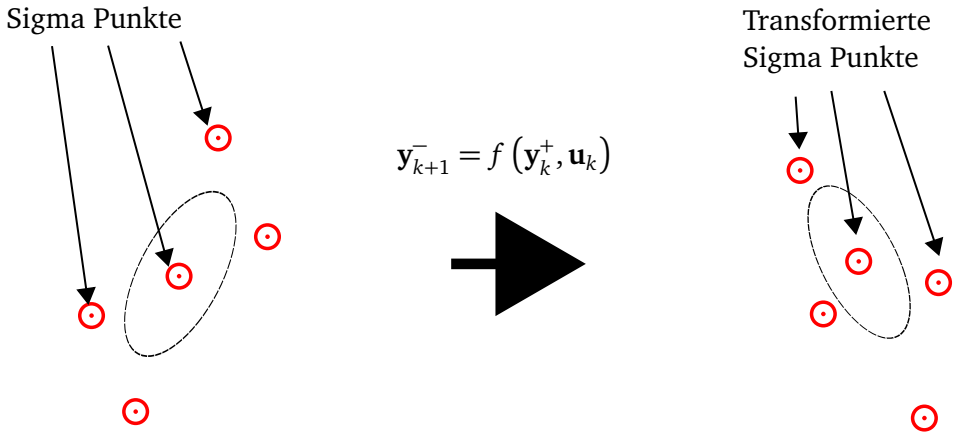


Abbildung 2.7.: 2D Beispiel für die approximierte Gaußverteilung nach einer *nichtlinearen* Transformation über Sigma Punkte. Nach einer Abbildung von Van Der Merwe (2004).

Für nichtlineare Zusammenhänge in einem Zustandsschätzproblem gibt es verschiedene Varianten des Kalman-Filters. Welche Methode letztendlich geeignet ist, hängt u. a. vom Grad der Nichtlinearität, der möglichen Abtastrate und der verfügbaren Rechenleistung ab. Dabei muss unterschieden werden zwischen einem nichtlinearen Systemmodell (nonlinear prediction) und einem nichtlinearen Zusammenhang zwischen Systemzustand und Messwerten (nonlinear fusion).

Die „*Arbeitspferde*“ (Levy, 1997) im Bereich Navigation sind das Extended Kalman-Filter (Gelb, 1974) sowie das linearisierte Kalman-Filter.

Darüber hinaus gibt es die Klasse der Sigma-Point Kalman-Filter (SPKF), vorgestellt von Julier et al. (1995). Ein Vorteil der Sigma-Point Kalman-Filter ist, dass die Vorhersage der Kovarianzmatrix keine Linearisierung von Gl. 2.106 benötigt. Die Vorhersage der Kovarianzmatrix erfolgt über eine Transformation von gewichteten Punkten (Sigma-Points), welche die ersten und zweiten Momente der Kovarianzmatrix beschreiben (Van Der Merwe und Wan, 2004). Für die Transformation der Punkte wird die eigentliche nichtlineare Vorhersagefunktion verwendet. Dieser Vorgang ist für ein 2D beispielhaft in Abbildung 2.7 gezeigt.

Ein SPKF kommt tendenziell mit stark nichtlinearen Modellen und längeren Vorhersagezeiträumen zu besseren Ergebnissen als ein EKF. Zu den SPKF zählen u. a.:

- Unscented Kalman-Filter (UKF)
- Central Difference Kalman-Filter (CDKF)
- Square-Root Unscented Kalman-Filter (SR-UKF)
- Square-Root Central Difference Kalman-Filter (SR-CDKF)

Der CDKF wurde unabhängig vom UKF entwickelt und ist in der Praxis gleichwertig zum UKF, siehe Van Der Merwe (2004). Die Square-Root (SR) Erweiterungen bestehen im Prinzip aus dem Einführen der QR- und Cholesky-Faktorisierung zur numerischen Robustifizierung, vgl. Abschnitt 2.8.

Ein Sigma-Point Kalman-Filter wird von Van Der Merwe und Wan (2004) erfolgreich zur Navigation eines UAVs (X-Cell-90 Helikopter) genutzt. Van Der Merwe und Wan (2004) heben hervor, dass die absoluten Fehler des Sigma-Point Kalman-Filters geringer sind als bei einem vergleichbaren EKF. Wendel et al. (2005) haben eine ähnliche Untersuchung durchgeführt und das EKF mit dem SPKF für die GPS/INS Fusion verglichen. Wendel et al. (2005) kommen dabei aber zu dem Ergebnis, dass für realistische Szenarien durch Einsatz eines Sigma-Point Kalman-Filters nicht mit einer lohnenswerten Verbesserung der Navigationsergebnissen zu rechnen ist bzw. die Ergebnisse beider Ansätze sind praktisch identisch.

Von Arasaratnam und Haykin (2009) wurde das Cubature Kalman-Filter (CKF) vorgeschlagen. Das CKF basiert auf einer Approximation zweiter Ordnung für nichtlineare Systeme. Zhao (2016) vergleicht im Rahmen einer GPS/INS Fusion ein Cubature Kalman-Filter mit einem EKF und kommt zu ähnlichen Ergebnissen wie bei dem genannten Vergleich zwischen SPKF und EKF.

2.9.1 Extended Kalman-Filter

Für nichtlineare Prozesse und/oder nichtlinearen Zusammenhängen zwischen den Beobachtungen und dem Zustandsvektor kann das normale Kalman-Filter aus Abschnitt 2.8 nicht oder nur zum Teil verwendet werden. Ähnlich wie bei einem nichtlinearen Ausgleichungsproblem (siehe Gl. 2.40) muss auf eine Reihenentwicklung nach Taylor zurückgegriffen werden. Nichtlineare Zusammenhänge werden in erster Ordnung approximiert (*First Order Filter*), Terme höherer Ordnung werden vernachlässigt. Dies erfordert die Berechnung der entsprechenden JACOBI-Matrix und setzt voraus, dass der Funktionsverlauf hinrei-

chend stetig ist und die Näherungswerte der Unbekannten ausreichend genau sind (Niemeier, 2008). Die Vorhersage kann streng mit dem eigentlichen nichtlinearen Zusammenhang erfolgen:

$$\mathbf{y}_k^- = f\left(\mathbf{y}_{k-1}^+, \mathbf{u}_{k-1}\right). \tag{2.113}$$

Die Fehlerfortpflanzung muss jedoch über eine JACOBI-Matrix $\mathbf{\Phi}_{k-1}$ näherungsweise durchgeführt werden:

$$\mathbf{\Phi}_{k-1} = \left.\frac{\partial f(\mathbf{y})}{\partial \mathbf{y}}\right|_{\mathbf{y}=\mathbf{y}_{k-1}^+} = \begin{pmatrix} \frac{\partial f_1}{\partial y_1} & \frac{\partial f_1}{\partial y_2} & \cdots & \frac{\partial f_1}{\partial y_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial y_1} & \frac{\partial f_m}{\partial y_2} & \cdots & \frac{\partial f_m}{\partial y_n} \end{pmatrix} \tag{2.114}$$

$$\mathbf{Q}_{yy,k}^- = \mathbf{\Phi}_{k-1} \cdot \mathbf{Q}_{yy,k-1}^+ \cdot \mathbf{\Phi}_{k-1}^T + \mathbf{Q}_{k-1}. \tag{2.115}$$

Je nach Grad der Nichtlinearität der funktionalen Zusammenhänge werden dadurch Terme höherer Ordnung nicht erfasst. Zu einem gewissen Teil kann die Unsicherheit auch dem Systemrauschen $\mathbf{Q}_{k-1}$ zugeschlagen werden, was allerdings eher eine Notlösung darstellt. Die negativen Effekte können aber minimiert werden, indem die Anzahl der Zwischenschritte erhöht wird. Das erhöht dafür den Rechenaufwand. Zudem müssen die JACOBI-Matrizen neu berechnet werden. Ab einem gewissen Punkt ist es daher sinnvoller ein SPKF einzusetzen.

Nichtlineare Messgleichungen werden im Prinzip gleichbehandelt wie eine nichtlineare Parameterschätzung. Die Residuen in Bezug auf den geschätzten Zustand können streng berechnet werden:

$$\mathbf{l} + \mathbf{v} = h(\mathbf{y}) \tag{2.116}$$

Der Innovationsschritt kann dabei (je nach gewünschter Genauigkeit) mehrfach wiederholt werden:

$$\mathbf{K}_k = \mathbf{Q}_{yy,k}^- \cdot \mathbf{A}_k^T \cdot \left(\mathbf{Q}_{ll,k} + \mathbf{A}_k \cdot \mathbf{Q}_{yy,k}^- \cdot \mathbf{A}_k^T\right)^{-1} \tag{2.117}$$

$$\mathbf{y}_k^+ = \mathbf{y}_k^- + \mathbf{K}_k \cdot \left(\mathbf{l}_k - h(\mathbf{y}_k^-)\right) \tag{2.118}$$

Die Designmatrix $\mathbf{A}$ entspricht der JACOBI-Matrix der nichtlinearen Messfunktion h:

$$\mathbf{A}_k = \left.\frac{\partial h(\mathbf{y})}{\partial \mathbf{y}}\right|_{\mathbf{y}=\mathbf{y}_k^-} = \begin{pmatrix} \frac{\partial h_1}{\partial y_1} & \frac{\partial h_1}{\partial y_2} & \cdots & \frac{\partial h_1}{\partial y_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial h_m}{\partial y_1} & \frac{\partial h_m}{\partial y_2} & \cdots & \frac{\partial h_m}{\partial y_n} \end{pmatrix}. \tag{2.119}$$

Nachdem das Abbruchkriterium erreicht wurde, wird mit der Jacobi-Matrix $\mathbf{A}_k$ aus dem letzten Iterationsschritt die Kovarianzmatrix $\mathbf{Q}_{yy}$ berechnet.

$$\mathbf{Q}^{+}_{yy,k} = (\mathbf{I} - \mathbf{K}_k \cdot \mathbf{A}_k) \cdot \mathbf{Q}^{-}_{yy,k}. \tag{2.120}$$

2.9.2 Linearisiertes Kalman-Filter

Im Gegensatz zum EKF werden im linearisierten Kalman-Filter nicht die absoluten Größen im Zustandsvektor $\mathbf{y}$ gesucht. Stattdessen wird zwar $\mathbf{y}$ mitgeführt, das linearisierte Kalman-Filter schätzt jedoch die Abweichungen $\delta\mathbf{y}$ in einem vermuteten Systemzustand $\hat{\mathbf{y}}$ (Wendel, 2011). Daher wird das linearisierte Kalman-Filter auch *Error-State* Kalman-Filter genannt (Farrell, 2008). Der Grund dafür ist, dass sich ein Fehler in einem vermuteten Systemzustand oft auf eine einfache und ggf. sogar komplett lineare Art entwickelt. In (Maybeck, 1982, Kap. 9.5) sowie Madyastha et al. (2011) werden EKF und linearisiertes Kalman-Filter vertieft behandelt und gegenübergestellt. Im Kern arbeiten beide Ansätze nach dem gleichen Prinzip, es ist eher eine Frage der Filtergestaltung, ob die absoluten Größen (*Total-Space*) oder die Abweichungen (*Error-State*) geschätzt werden sollen. Für die Navigationszustandsschätzung beispielsweise sind linearisierte Kalman-Filter beliebt, siehe u. a. Maybeck (1979), Grewal und Andrews (2001), Titterton und Weston (2004), Farrell (2008) oder Wendel (2011).

Im Grund lassen sich, wie bereits in Abschnitt 2.8 beschrieben, diese Arten der Zustandsschätzung auf eine einheitliche Basis zusammenfassen. Das beinhaltet auch die robusten M-Schätzer, wie z. B. die $\mathscr{L}_1$-Norm, die auf dieser Grundlage behandelt werden können. Im Folgenden soll gezeigt werden, dass auch *Error-State* Kalman-Filter auf diese gemeinsame Basis zurückzuführen sind.

Dazu gilt **1.)** für den Linearisierungspunkt $\mathbf{y}^{+}_{k,0}$:

$$\mathbf{y}^{-}_{k} + \mathbf{v}_{y^{-}_{k}} = \mathbf{I} \cdot \delta\mathbf{y}^{+}_{k} + \mathbf{y}^{+}_{k,0} \tag{2.121}$$

und **2.)**:

$$\mathbf{l}_k + \mathbf{v}_{l_k} = \mathbf{A}(\mathbf{y}^{+}_{k,0}) \cdot \delta\mathbf{y}^{+}_{k} + \mathbf{l}(\mathbf{y}^{+}_{k,0}). \tag{2.122}$$

Übrig bleibt noch die Komponente der Vorhersage **3.)** (siehe Chapman-Kolmogorov-Gl., 2.7):

$$\mathbf{y}^{-}_{k} = \mathbf{y}^{-}_{k}(\mathbf{y}^{+}_{k-1}). \tag{2.123}$$

Komponente **3.** kann für sich alleine betrachtet werden, besonders im Hinblick auf die genannten Error-State Zustandsschätzer. Da hier das zeitliche Verhalten von Fehlern betrachtet wird, basieren die Zusammenhänge oft auf Differentialgleichungen, wie z. B. $\dot{\Omega} = \mathbf{R} \cdot \Omega$. Somit kann Gl. 2.123 durchaus die Lösung von Differentialgleichungen enthalten. Letztendlich ist $\mathbf{y}^{-}_{k}$ (**1.)**) wieder eine Eingabe für **3.)**.

Oft wird in der Literatur (z. B. Farrell (2008) oder Wendel (2011)) dazu das Zustandsübergangsverhalten in folgender Form betrachtet (kompatibel zu den vorangehenden Überlegungen):

$$\dot{\mathbf{y}} = f(\mathbf{y}) + \mathbf{B} \cdot \mathbf{u} + \mathbf{G} \cdot \mathbf{w} \tag{2.124}$$

Darüberhinaus sei für diese System auch ein genäherter bzw. nominaler Zustandsvektor an einem Linearisierungspunkt $\mathbf{y}_0$ gegeben (Maybeck, 1979), (Farrell, 2008):

$$\dot{\mathbf{y}}_0 = f(\mathbf{y}_0) + \mathbf{B} \cdot \mathbf{u} \tag{2.125}$$

$$\mathbf{l}_0 = h(\mathbf{y}_0). \tag{2.126}$$

Bildet man nun die Differenz zwischen Gl. 2.124 und 2.125:

$$\delta\dot{\mathbf{y}} = \dot{\mathbf{y}} - \dot{\mathbf{y}}_0 = f(\mathbf{y}) - f(\mathbf{y}_0) + \mathbf{G} \cdot \mathbf{w}. \tag{2.127}$$

Dies lässt sich als eine Approximation mit Hilfe einer Taylorreihenentwicklung unter der Vernachlässigung von Termen höherer Ordnung schreiben als:

$$\dot{\mathbf{y}} - \dot{\mathbf{y}}_0 \approx f(\mathbf{y}_0) + \left.\frac{\partial f(\mathbf{y})}{\partial \mathbf{y}}\right|_{\mathbf{y}=\mathbf{y}_0} \cdot (\mathbf{y} - \mathbf{y}_0) - f(\mathbf{y}_0) + \mathbf{G} \cdot \mathbf{w} \tag{2.128}$$

Diese Gleichung vereinfacht sich zu:

$$\dot{\mathbf{y}} - \dot{\mathbf{y}}_0 = \left.\frac{\partial f(\mathbf{y})}{\partial \mathbf{y}}\right|_{\mathbf{y}=\mathbf{y}_0} \cdot (\mathbf{y} - \mathbf{y}_0) + \mathbf{G} \cdot \mathbf{w} \tag{2.129}$$

Die Abweichung zwischen dem Systemzustand $\mathbf{y}$ und dem nominalen Systemzustand $\mathbf{y}_0$ sei der Fehlervektor $\delta\mathbf{y}$:

$$\delta\mathbf{y} = \mathbf{y} - \mathbf{y}_0. \tag{2.130}$$

Zudem wird als Entwicklungspunkt der aktuell geschätzte Zustand $\hat{\mathbf{y}}$ gewählt:

$$\delta\dot{\mathbf{y}} = \left.\frac{\partial f(\mathbf{y})}{\partial \mathbf{y}}\right|_{\mathbf{y}=\hat{\mathbf{y}}} \cdot \delta\mathbf{y} + \mathbf{G} \cdot \mathbf{w}. \tag{2.131}$$

Dadurch wird in erster Ordnung das zeitliche Verhalten von einem Schätzfehler beschrieben (Wendel, 2011). Maybeck (1979) nennt Gl. 2.131 die *linearized perturbation equation*. Die Kernidee bzw. die Voraussetzung ist, dass das zeitliche Verhalten eines Schätzfehlers sich näherungsweise linear verhält

und damit auch mit dem Gaußschen Fehlerfortpflanzungsgesetz gut vorhersagen lässt. Die JACOBI-Matrix dazu wird üblicherweise als **F** bezeichnet:

$$\mathbf{F} = \left.\frac{\partial f(\mathbf{y})}{\partial \mathbf{y}}\right|_{\mathbf{y}=\hat{\mathbf{y}}} = \begin{pmatrix} \frac{\partial f_1}{\partial y_1} & \frac{\partial f_1}{\partial y_2} & \cdots & \frac{\partial f_1}{\partial y_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial y_1} & \frac{\partial f_m}{\partial y_2} & \cdots & \frac{\partial f_m}{\partial y_n} \end{pmatrix} \tag{2.132}$$

$$\delta\dot{\mathbf{y}} = \mathbf{F} \cdot \delta\mathbf{y} + \mathbf{G} \cdot \mathbf{w}. \tag{2.133}$$

Das linearisierte Kalman-Filter schätzt den Fehler in der Zustandsschätzung; mit jedem Korrekturschritt wird der *außerhalb* mitgeführte geschätzte Zustandsvektor $\hat{\mathbf{y}}$ entsprechend korrigiert, danach ist $\delta\mathbf{y}^+$ nach bestem Wissen gleich 0, daher ist folgender Schritt *nicht* notwendig:

$$\delta\hat{\mathbf{y}}_k^- = \mathbf{\Phi}_{k-1} \cdot \delta\hat{\mathbf{y}}_{k-1}^+. \tag{2.134}$$

Der Vektor $\delta\hat{\mathbf{y}}$ darf sich in der Parametrisierung von $\hat{\mathbf{y}}$ unterscheiden[12]. Dennoch muss der geschätzte Zustandsvektor $\hat{\mathbf{y}}$ entsprechend der Gl. 2.124 „*manuell*“ propagiert werden. Hier können verschiedene numerische Verfahren zur Lösung von Differentialgleichungen zum Einsatz kommen. Im Idealfall kann eine geschlossene Lösung gefunden werden. Ansonsten bieten sich Verfahren wie z. B das von Runge-Kutta an (Runge, 1895). Die Fortführung des stochastischen Modells verläuft analog zum linearen Kalman-Filter, vgl. Gl. 2.106:

$$\mathbf{Q}_{yy,k}^- = \mathbf{\Phi}_{k-1} \cdot \mathbf{Q}_{yy,k-1}^+ \cdot \mathbf{\Phi}_{k-1}^T + \mathbf{Q}_{k-1}. \tag{2.135}$$

Um zu der Transitionsmatrix $\mathbf{\Phi}$ zu gelangen, muss über ein Zeitintervall t (von Epoche $k-1$ zu k) die Differentialgleichung 2.131 gelöst werden[13]. Unter der Annahme, dass sich **F** in diesem Intervall nicht ändert, gilt allgemein (Grewal und Andrews, 2001):

$$\mathbf{\Phi} = e^{\mathbf{F} \cdot t} \tag{2.136}$$

$$\mathbf{\Phi} = \mathbf{I} + \mathbf{F} \cdot t + \frac{1}{2}(\mathbf{F} \cdot t)^2 + \frac{1}{3!}(\mathbf{F} \cdot t)^3 \ldots \tag{2.137}$$

Nach Untersuchungen von Petovello (2003) ist es auch bei hohen Genauigkeitsanforderungen (cm-Genauigkeit) nicht notwendig über die zweite Ordnung in Gl. 2.137 hinauszugehen. Falls sich **F** doch zu stark in dem Intervall t ändert, empfiehlt Farrell (2008) den Zeitraum gleichmäßig so zu teilen, dass es

[12] Beispielsweise kann eine Orientierung in diesem absoluten Zustandsvektor als Quaternion (4 × 1 Vektor) gespeichert werden, während der Zustandsvektor des linearisierten Kalman-Filters eine 3 × 1 Winkelfehlerdarstellung enthält, siehe Abschnitt 4.1. Für die Navigation mit einer Strapdown IMU sei auf Abschnitt 3.3 verwiesen.

[13] Die zur Aufstellung von Gleichung 2.135 benötigte **F** Matrix für die Navigationszustandsschätzung wird in Abschnitt 4.1 vertieft behandelt.

annehmbar ist, $\mathbf{F}$ als konstant anzunehmen. Dann muss $\mathbf{\Phi}$ über das Intervall hinweg Schritt für Schritt verkettet werden:

$$\mathbf{\Phi}_{t_{i+1},t_0} = \mathbf{\Phi}_{t_{i+1},t_i} \cdot \mathbf{\Phi}_{t_i,t_0} \tag{2.138}$$

In der Zustandsraumdarstellung von linearen Systemen in Gl. 2.124 modelliert $\mathbf{w}$ das Systemrauschen im zeitkontinuierlichen Bereich. Wendel (2011) zeigt, dass wenn die Matrix $\mathbf{Q}$ die spektrale Leistungsdichte des Rauschvektors $\mathbf{w}$ beschreibt, dann kann die Fehlerfortpflanzung im Zeitdiskreten (bei einer Zeitschrittweite t) ersatzweise mit einer Matrix $\mathbf{Q}_k$ approximiert werden:

$$\mathbf{Q}_k = \mathbf{G} \cdot \mathbf{Q} \cdot \mathbf{G}^T \cdot t. \tag{2.139}$$

Abschnitt 2.10 und 2.11 gehen im Detail darauf ein, wie $\mathbf{Q}$ für bestimmte Sensoren bestimmt werden kann.

Im **Korrekturschritt** muss im Sinne einer vermittelnden Ausgleichung zuerst ein formelmäßiger Zusammenhang zwischen dem Messvektor $\mathbf{l}$ und dem Systemzustand $\mathbf{y}$ hergestellt werden.

$$\mathbf{l} + \mathbf{v} = h(\mathbf{y}). \tag{2.140}$$

Aus dem geschätzten a priori Systemzustand lässt sich eine erwartete Messung berechnen:

$$\hat{\mathbf{l}} = h(\hat{\mathbf{y}}^-). \tag{2.141}$$

Die Differenz von der erwarteten Messungen zu der tatsächlichen Messung $\mathbf{l}$ führt dann zum Messvektor $\delta\mathbf{l}$ für das linearisierte Kalman-Filter:

$$\delta\mathbf{l} = \mathbf{l} - h(\hat{\mathbf{y}}^-) = \mathbf{l} - \hat{\mathbf{l}}. \tag{2.142}$$

Auf Grundlage von Gleichung 2.145 kann dann mit den üblichen Kalman-Filter Gleichungen gearbeitet werden, siehe Abschnitt 2.8. Die Designmatrix $\mathbf{A}$ ist die Jacobi-Matrix der Funktion h.

$$\delta\mathbf{l} + \mathbf{v} = \left.\frac{\partial h(\mathbf{y})}{\partial \mathbf{y}}\right|_{\mathbf{y}=\hat{\mathbf{y}}} \cdot \delta\mathbf{y} \tag{2.143}$$

$$\mathbf{A} = \left.\frac{\partial h(\mathbf{y})}{\partial \mathbf{y}}\right|_{\mathbf{y}=\hat{\mathbf{y}}} = \begin{pmatrix} \frac{\partial h_1}{\partial y_1} & \frac{\partial h_1}{\partial y_2} & \cdots & \frac{\partial h_1}{\partial y_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial h_m}{\partial y_1} & \frac{\partial h_m}{\partial y_2} & \cdots & \frac{\partial h_m}{\partial y_n} \end{pmatrix} \tag{2.144}$$

$$\delta\mathbf{l} + \mathbf{v} = \mathbf{A} \cdot \delta\mathbf{y}. \tag{2.145}$$

Das Vorgehen ist damit gleich wie bei einem Extended Kalman-Filter, siehe Abschnitt 2.9.1.

Beispiel: Im Folgenden soll ein Beispiel für ein lineares Kalman-Filter gezeigt werden. Im Hinblick auf die Navigation in Kapitel 4 soll dazu eine Lage geschätzt werden. Dieses Beispiel soll die Sonderstellung der Error-State Formulierung in der Navigation hervorheben und greift dabei auf Definitionen von Abschnitt 3.1 vor.

Dem linearisierten Kalman-Filter entsprechend werden die Lagefehler $\delta\phi$, $\delta\theta$ und $\delta\psi$ geschätzt. Zusammengefasst zu dem Vektor $\boldsymbol{\psi} = \begin{pmatrix}\delta\phi & \delta\theta & \delta\psi\end{pmatrix}^T$. Die Fehler $\delta\phi$ und $\delta\theta$ entsprechen einem Fehler im Roll- u. Pitch-Winkel, der Fehler $\delta\psi$ entspricht einem Azimutfehler. Der Grund für diese Darstellung ist, dass daraus (wie im Folgenden gezeigt) letztendlich einfache Zusammenhänge resultieren (Titterton und Weston, 2004, Kap. 12.3.1.1), (Farrell, 2008, Kap. 10.5.2.2), (Wendel, 2011, Kap. 8.2). Aus den kleinen Winkelfehlern kann eine schiefsymmetrische Matrix gebildet werden:

$$\boldsymbol{\Psi} = [\boldsymbol{\psi}\times] = \begin{pmatrix} 0 & -\delta\psi & \delta\theta \\ \delta\psi & 0 & -\delta\phi \\ -\delta\theta & \delta\phi & 0 \end{pmatrix}. \tag{2.146}$$

Zudem soll ein Biasfehler in den Drehratenmessungen als Hilfsparameter[14] mitgeschätzt werden: $\delta\mathbf{b}_\omega$. Der Zustandsvektor $\delta\mathbf{y}$ hat folgende Form:

$$\delta\mathbf{y} = \begin{pmatrix}\delta\phi & \delta\theta & \delta\psi & \delta b_{\omega,x} & \delta b_{\omega,y} & \delta b_{\omega,z}\end{pmatrix}^T = \begin{pmatrix}\boldsymbol{\psi}^T & \delta\mathbf{b}_\omega{}^T\end{pmatrix}^T \tag{2.147}$$

Im weiteren Verlauf sollen nun die formelmäßigen Zusammenhänge für den Vorhersageschritt hergeleitet werden (vgl. Gl. 2.133):

$$\delta\dot{\mathbf{y}} = \mathbf{F}\cdot\delta\mathbf{y}. \tag{2.148}$$

Die Grundlage dafür bietet die Differentialgleichung für Drehmatrizen:

$$\dot{\mathbf{R}}_b^n = \mathbf{R}_b^n \cdot \boldsymbol{\Omega}_{nb}^b \tag{2.149}$$

$$= \mathbf{R}_b^n \cdot \left[\boldsymbol{\omega}_{nb}^b \times\right]. \tag{2.150}$$

[14] Engl.: *nuisance parameter* genannt. Parameter, die eigentlich nicht von Interesse sind, aber trotzdem notwendig sind um den Systemzustand zu beschreiben (Grewal und Andrews, 2001).

Der Zusammenhang zwischen einer geschätzten Drehmatrix $\hat{\mathbf{R}}_b^n$ und der wahren Drehmatrix $\mathbf{R}_b^n$ lässt sich über $\boldsymbol{\psi}$ beschreiben (Titterton und Weston, 2004, Kap. 12.3.1.1):

$$\hat{\mathbf{R}}_b^n = (\mathbf{I} - \boldsymbol{\Psi}) \cdot \mathbf{R}_b^n \tag{2.151}$$

$$\hat{\mathbf{R}}_n^b = \mathbf{R}_n^b \cdot (\mathbf{I} + \boldsymbol{\Psi}) \tag{2.152}$$

$$\mathbf{R}_n^b = \hat{\mathbf{R}}_n^b \cdot (\mathbf{I} - \boldsymbol{\Psi}) \tag{2.153}$$

$$\mathbf{R}_b^n = (\mathbf{I} + \boldsymbol{\Psi}) \cdot \hat{\mathbf{R}}_b^n \tag{2.154}$$

Leitet man nun den rechten Teil von Gl. 2.154 mit Hilfe der Produktregel ab

$$\frac{\mathrm{d}}{\mathrm{d}t}(\mathbf{I} + \boldsymbol{\Psi}) \cdot \hat{\mathbf{R}}_b^n = \dot{\boldsymbol{\Psi}} \cdot \hat{\mathbf{R}}_b^n + (\mathbf{I} + \boldsymbol{\Psi}) \cdot \dot{\hat{\mathbf{R}}}_b^n, \tag{2.155}$$

so kann dieser Teil in Gl. 2.149 eingesetzt werden:

$$\dot{\boldsymbol{\Psi}} \cdot \hat{\mathbf{R}}_b^n + (\mathbf{I} + \boldsymbol{\Psi}) \cdot \dot{\hat{\mathbf{R}}}_b^n = \mathbf{R}_b^n \cdot \boldsymbol{\Omega}_{nb}^b. \tag{2.156}$$

Anschließend führt man in Gl. 2.156 auch auf der rechten Seite Gl. 2.154 ein:

$$\dot{\boldsymbol{\Psi}} \cdot \hat{\mathbf{R}}_b^n + (\mathbf{I} + \boldsymbol{\Psi}) \cdot \dot{\hat{\mathbf{R}}}_b^n = (\mathbf{I} + \boldsymbol{\Psi}) \cdot \hat{\mathbf{R}}_b^n \cdot \boldsymbol{\Omega}_{nb}^b. \tag{2.157}$$

Die Drehrate des Objekts gegenüber dem Navigationskoordinatensystem (siehe dazu auch Abschnitt 3.1) $\boldsymbol{\Omega}_{nb}^b$ lässt sich aufteilen in die Drehrate gegenüber dem Inertialsystem $\boldsymbol{\Omega}_{ib}^b$ und der Drehrate des Koordinatensystems gegenüber dem Inertialsystem $\boldsymbol{\Omega}_{in}^b$:

$$\boldsymbol{\Omega}_{nb}^b = \boldsymbol{\Omega}_{ib}^b - \boldsymbol{\Omega}_{in}^b. \tag{2.158}$$

Somit erweitert sich Gl. 2.157 zu:

$$\dot{\boldsymbol{\Psi}} \cdot \hat{\mathbf{R}}_b^n + (\mathbf{I} + \boldsymbol{\Psi}) \cdot \dot{\hat{\mathbf{R}}}_b^n = (\mathbf{I} + \boldsymbol{\Psi}) \cdot \hat{\mathbf{R}}_b^n \cdot \left(\boldsymbol{\Omega}_{ib}^b - \boldsymbol{\Omega}_{in}^b\right). \tag{2.159}$$

Diese Darstellung wird nun erweitert um eine Abweichung zwischen den wahren Drehraten und den geschätzten Drehraten:

$$\delta\boldsymbol{\Omega}_{ib}^b = \boldsymbol{\Omega}_{ib}^b - \hat{\boldsymbol{\Omega}}_{ib}^b \tag{2.160}$$

$$\delta\boldsymbol{\Omega}_{in}^b = \boldsymbol{\Omega}_{in}^b - \hat{\boldsymbol{\Omega}}_{in}^b. \tag{2.161}$$

Daraus lässt sich nach Farrell (2008) folgende Umformung durchführen:

$$\dot{\boldsymbol{\Psi}} \cdot \hat{\mathbf{R}}_b^n + (\mathbf{I} + \boldsymbol{\Psi}) \cdot \dot{\hat{\mathbf{R}}}_b^n = (\mathbf{I} + \boldsymbol{\Psi}) \cdot \hat{\mathbf{R}}_b^n \cdot \left(\hat{\boldsymbol{\Omega}}_{ib}^b - \hat{\boldsymbol{\Omega}}_{in}^b + \delta\boldsymbol{\Omega}_{ib}^b - \delta\boldsymbol{\Omega}_{in}^b \right) \tag{2.162}$$

$$= (\mathbf{I} + \boldsymbol{\Psi}) \cdot \dot{\hat{\mathbf{R}}}_b^n + (\mathbf{I} + \boldsymbol{\Psi}) \cdot \hat{\mathbf{R}}_b^n \cdot \left(\delta\boldsymbol{\Omega}_{ib}^b - \delta\boldsymbol{\Omega}_{in}^b \right) \tag{2.163}$$

Üblicherweise, siehe Titterton und Weston (2004), Farrell (2008) oder Wendel (2011), werden nun die Produkte aus zwei Fehlern vernachlässigt: $\boldsymbol{\Psi} \cdot \hat{\mathbf{R}}_b^n \cdot \left(\delta\boldsymbol{\Omega}_{ib}^b - \delta\boldsymbol{\Omega}_{in}^b \right)$, somit vereinfacht sich Gl. 2.163 zu:

$$\dot{\boldsymbol{\Psi}} = \left[\dot{\boldsymbol{\psi}} \times \right] = \hat{\mathbf{R}}_b^n \cdot \left(\delta\boldsymbol{\Omega}_{ib}^b - \delta\boldsymbol{\Omega}_{in}^b \right) \cdot \hat{\mathbf{R}}_n^b. \tag{2.164}$$

Folgender allgemeiner Zusammenhang für kreuzproduktbildende Matrizen kann für eine weitere Vereinfachung von Gl. 2.164 genutzt werden:

$$\boldsymbol{\Omega}^n = \mathbf{R}_b^n \cdot \boldsymbol{\Omega}^b \cdot \mathbf{R}_n^b. \tag{2.165}$$

Somit lässt sich Gl. 2.164 in eine Vektorform bringen:

$$\dot{\boldsymbol{\Psi}} = \delta\boldsymbol{\Omega}_{ib}^n - \delta\boldsymbol{\Omega}_{in}^n \tag{2.166}$$

$$\dot{\boldsymbol{\psi}} = \delta\boldsymbol{\omega}_{ib}^n - \delta\boldsymbol{\omega}_{in}^n = \hat{\mathbf{R}}_b^n \cdot \left(\delta\boldsymbol{\omega}_{ib}^b - \delta\boldsymbol{\omega}_{in}^b \right). \tag{2.167}$$

Um dieses Beispiel einzugrenzen, soll an dieser Stelle die Drehrate $\boldsymbol{\omega}_{in}^n$ vernachlässigt werden[15], dies ist für ein reines Filter zur Lageschätzung durchaus üblich, bzw. ohne Positionsinformation gar nicht anders möglich (Farrell, 2008), (Wendel, 2011). Nun wird angenommen, dass der Fehler in der Drehrate $\delta\boldsymbol{\omega}_{ib}^b$ aus einem Fehler im Gyroskop Bias resultiert:

$$\delta\boldsymbol{\omega}_{ib}^b = -\delta\mathbf{b}_\omega = \boldsymbol{\omega}_{ib}^b - \hat{\boldsymbol{\omega}}_{ib}^b. \tag{2.168}$$

Somit kann letztendlich die zeitkontinuierliche Transitionsgleichung 2.133 formuliert werden:

$$\dot{\boldsymbol{\psi}} = -\hat{\mathbf{R}}_b^n \cdot \delta\mathbf{b}_\omega. \tag{2.169}$$

Gleichung 2.169 kann nun direkt für den Aufbau der JACOBI-Matrix $\mathbf{F}$ in Gl. 2.133 genutzt werden:

$$\underbrace{\begin{pmatrix} \dot{\boldsymbol{\psi}} \\ \delta\dot{\mathbf{b}}_\omega \end{pmatrix}}_{\delta\dot{\mathbf{y}}} = \underbrace{\begin{pmatrix} \mathbf{0}_{3\times3} & -\hat{\mathbf{R}}_b^n \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \end{pmatrix}}_{\mathbf{F}} \cdot \underbrace{\begin{pmatrix} \boldsymbol{\psi} \\ \delta\mathbf{b}_\omega \end{pmatrix}}_{\delta\mathbf{y}} + \underbrace{\begin{pmatrix} \hat{\mathbf{R}}_b^n & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{I}_{3\times3} \end{pmatrix}}_{\mathbf{G}} \cdot \underbrace{\begin{pmatrix} \mathbf{n}_\omega \\ \mathbf{n}_{b_\omega} \end{pmatrix}}_{\mathbf{w}} \tag{2.170}$$

[15] Gl. 2.167 lautet vollständig: $\dot{\boldsymbol{\psi}} = \mathbf{R}_b^n \cdot \delta\boldsymbol{\omega}_{ib}^b - \delta\boldsymbol{\omega}_{in}^n - \boldsymbol{\Omega}_{in}^n \cdot \boldsymbol{\psi}$, siehe z. B. (Wendel, 2011, Kap. 8.2).

Für die vollständige Vorhersage muss Gl. 2.170 mit Gl. 2.137 zu einer Transitionsmatrix $\mathbf{\Phi}_k$ aufintegriert werden. Für die Stützung dieses Filters durch Beobachtungsgleichungen sei auf Abschnitt 4.5.1 verwiesen. Der Vektor $\mathbf{w}$ enthält das Gyroskoprauschen $\mathbf{n}_\omega$ sowie den Bias Random Walk Parametervektor $\mathbf{n}_{b_\omega}$. Diese Größen müssen individuell für jeden IMU Typ bestimmt werden. Die Bestimmung ist Gegenstand von Abschnitt 2.10 und 2.11. Das zeitdiskrete Systemrauschen $\mathbf{Q}_k$ kann mit Gl. 2.139 berechnet werden. Die Vorhersage der Orientierung wird außerhalb des Kalman-Filters mitgeführt. Beispielsweise in Form eines Lagequaternions, siehe Gl. 3.19.

2.10 Allan Deviation

Vor dem Hintergrund der vorgestellten Verfahren der Zustandsschätzung wird eine Methode benötigt, um das stochastische Verhalten von Systemen und Sensoren zu beschreiben. Der Ausgangspunkt ist dabei das Systemrauschen $\mathbf{w}$ im zeitkontinuierlich Bereich, vgl. Gl. 2.124 (hier für lineare Zusammenhänge):

$$\dot{\mathbf{y}} = f\left(\mathbf{y}\right) + \mathbf{B} \cdot \mathbf{u} + \mathbf{G} \cdot \mathbf{w}. \tag{2.171}$$

Unter der Annahme, dass die Verteilung im zeitkontinuierlich Bereich durch eine Normalverteilung beschrieben werden kann:

$$\mathbf{w} \sim \mathcal{N}(0, \mathbf{Q}) \tag{2.172}$$

Dieser Abschnitt beschäftigt sich im Weiteren mit der Bestimmung von $\mathbf{Q}$ für Inertialsensoren. Diese Matrix enthält zum einen die spektrale Leistungsdichte des Systemrauschens, zum anderen kann in dieser Matrix auch der sogenannte Bias Random Walk modelliert werden. Dazu wird häufig die sogenannte *Allan Varianz* genutzt (Farrell, 2008). Die *Allan Deviation* (AD) ist die Wurzel der Allan Varianz (AVAR). Ursprünglich von Allan (1966) zur Bewertung der Stabilität und dem Rauschverhalten von Atomuhren entwickelt, lässt sich die Allan Varianz sehr gut auf Gyroskope und Beschleunigungsmesser anwenden. Die Grundidee ist, dass man Messwerte von einem ruhenden Sensor (da sonst nicht zwischen Bewegung, Rauschen und Bias unterschieden werden kann) über längere Zeit aufnimmt und dann in m Abschnitte bzw. Behälter (engl. *bins* oder *cluster*) mit der Länge τ einteilt. Für jeden Zeitabschnitt i der Länge τ wird dann der Mittelwert $a(\tau)_i$ gebildet. Die Hälfte der mittleren Abweichung der Quadrate der aufeinanderfolgenden Differenzen ist dann die Allan Varianz für die Abschnittslänge τ. Als Formel ausgedrückt:

$$\mathrm{AVAR}(\tau) = \frac{1}{2} \cdot \frac{1}{(m-1)} \sum_{i=0}^{m-2} \Big(a(\tau)_{i+1} - a(\tau)_i \Big)^2. \tag{2.173}$$

Durch den Faktor $\frac{1}{2}$ entspricht die Allan Varianz bei zufälligen und unkorrelierten Daten der klassischen Varianz (Allan et al., 1997).

Ein Gyroskop mit einem weißen Rauschen von $\sigma_V = 0,005 \frac{\text{rad/s}}{\sqrt{\text{Hz}}}$ und einem Bias Random Walk[16] mit der Stärke $\sigma_b = 0,002 \frac{\text{rad/s}}{\sqrt{\text{s}}}$ ergibt eine Allan Deviationskurve, die in Abbildung 2.8 zu sehen ist. Bei kurzen Zeitabschnitten (τ) dominiert das weiße Rauschen, sodass an der Stelle AD(1 s) direkt σ_V abzulesen ist ($\hat{\sigma}_V = 0,005097 \frac{\text{rad/s}}{\sqrt{\text{Hz}}}$). Die Bias Stabilität (BS) (ein wichtiges Gütekriterium für ein Gyroskop) ist definiert als der tiefste Punkt in der Allan Deviationskurve. In Abbildung 2.8 liegt die Bias Stabilität bei AD(4 s) = 0,0034 rad/s. Um den Bias Random Walk Parameter aus der Kurve zu entnehmen ist ein höherer Aufwand notwendig. Da dieser Parameter für die späteren Kapitel wichtig ist, soll hier kurz die Vorgehensweise vorgestellt werden. Das Verfahren ist von der IEEE (2006) Spezifikation übernommen. Die Allan Varianz lässt sich (über die spektrale Leistungsdichte) mit folgenden zugrundeliegenden zufälligen Driftkomponenten ausdrücken (Gl. 2.174). Als Quelle (und Herleitung) sei wieder auf die IEEE (2006) Veröffentlichung verwiesen.

$$\text{AVAR}(\tau) = \frac{R^2 \cdot \tau^2}{2} + \frac{K^2 \cdot \tau}{3} + B^2 \cdot \left(\frac{2}{\pi}\right) \cdot \ln(2) + \frac{N^2}{\tau} + \frac{3 \cdot Q^2}{\tau^2} \tag{2.174}$$

Diese Darstellung nimmt an, dass sich das Rauschverhalten aus den folgenden Einflüssen zusammensetzt:

- R - Rampen Koeffizient. Ein systematischer Einfluss, der linear von der Zeit abhängig ist. Z. B. bei Gyroskopen: $\omega = R \cdot t$.
- K - Bias Random Walk Koeffizient (die hier gesuchte Größe für die stochastische Modellierung).
- B - Bias Instability Koeffizient.
- N - Wurzel der spektralen Leistungsdichte. Dieser Wert entspricht (im Rahmen der möglichen Genauigkeit) dem Wert der Allan Deviation bei $\tau = 1$ Sekunde.
- Q - Rauschen, das durch Quantisierung entsteht. Dieser Einfluss ist abhängig von der Sensorkonfiguration (Anzahl der Bits für das Auslesen der Sensorwerte).

$$\begin{pmatrix} \text{AVAR}(\tau_1) \\ \text{AVAR}(\tau_2) \\ \cdots \\ \text{AVAR}(\tau_n) \end{pmatrix} = \begin{pmatrix} \frac{\tau_1^2}{2} & \frac{\tau_1}{3} & \frac{2}{\pi}\ln(2) & \frac{1}{\tau_1} & \frac{3}{\tau_1^2} \\ \frac{\tau_2^2}{2} & \frac{\tau_2}{3} & \frac{2}{\pi}\ln(2) & \frac{1}{\tau_2} & \frac{3}{\tau_2^2} \\ & & \cdots & & \\ \frac{\tau_n^2}{2} & \frac{\tau_n}{3} & \frac{2}{\pi}\ln(2) & \frac{1}{\tau_n} & \frac{3}{\tau_n^2} \end{pmatrix} \cdot \begin{pmatrix} R^2 \\ K^2 \\ B^2 \\ N^2 \\ Q^2 \end{pmatrix} \tag{2.175}$$

Mit Hilfe von Gleichung 2.175 lassen sich die gesuchten Größen mit Hilfe der Methode der kleinsten Quadrate schätzen. Dabei ist es günstig, die Quadrate der gesuchten Größen zu schätzen, um bei einem linearen Ausgleichungsmodell zu bleiben. Hinzu kommt, dass es sinnvoll sein kann, die einzelnen Messungen zu gewichten. Das kommt daher, dass die Allan Varianzen mit großem Abtastintervall eine

[16] Nach Gelb (1974): Der Random Walk Prozess resultiert, wenn unkorrelierte Signale aufintegriert werden.

höhere Ungenauigkeit haben, da mit zunehmender Abtastlänge τ weniger Messungen zur Verfügung stehen. Ein Ansatz wäre z. B. die Gewichtung anhand der zur Verfügung stehenden Abtastintervalle (m_i). Die Gewichtsmatrix $\mathbf{P}$ für die Ausgleichungsaufgabe wäre dann:

$$\mathbf{P} = \begin{pmatrix} \frac{1}{\sqrt{m_1}} & & & \\ & \frac{1}{\sqrt{m_2}} & & \\ & & \ddots & \\ & & & \frac{1}{\sqrt{m_n}} \end{pmatrix}. \tag{2.176}$$

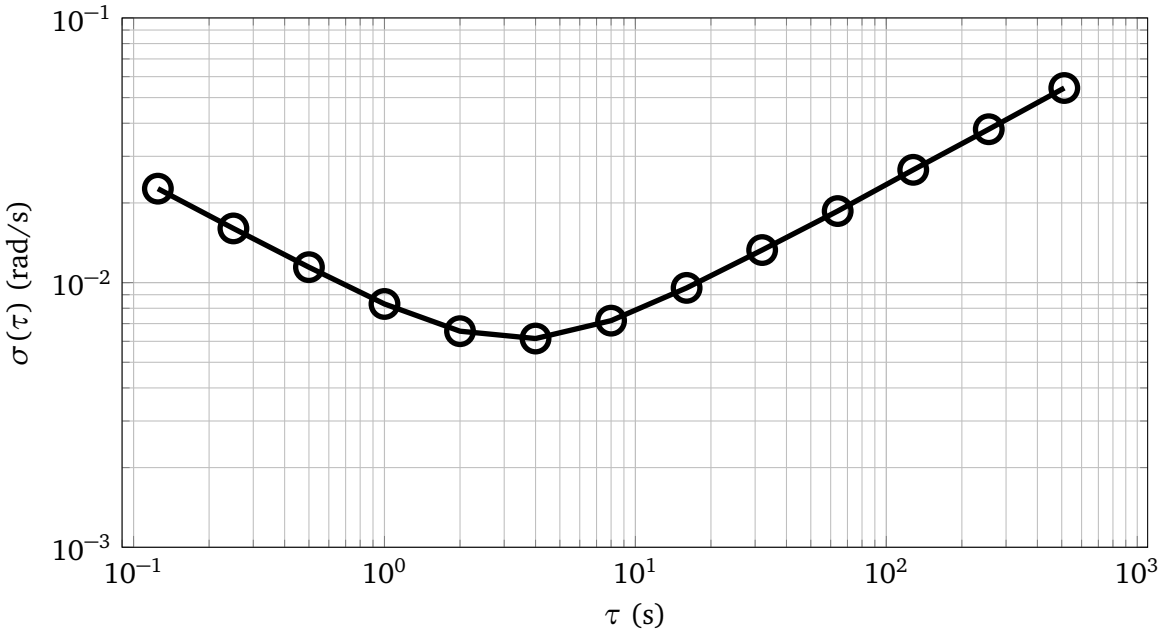


Abbildung 2.8.: Allan Deviation von Gyroskop-Messungen.

2.11 Stochastische Modellierung von IMU Sensorfehlern

Um das Rauschverhalten von IMUs (also Beschleunigungsmesser und Gyroskope) zu modellieren, bietet sich die spektrale Leistungsdichte des Systemrauschens (engl. *noise spectral density*) an. Dabei wird an dieser Stelle angenommen, dass es sich bei dem Sensorrauschen vornehmlich um weißes Rauschen handelt. Also ein mittelwertfreies Signal, das mit sich selbst nicht korreliert ist. Die spektrale Leistungsdichte des Systemrauschens berechnet sich aus der Fouriertransformation der Autokorrelationsfunktion des verrauschten Signals (Hänsler, 2013). Im Frequenzbereich sieht man dem Signal an, wie stark es in welchen Frequenzbereichen variiert. Ein diskreter (weißer) Rauschanteil eines integrierenden Sensors

zu einer Epoche k (v_k) bildet sich aus einem zeitkontinuierlichen Rauschen $V(t)$ über ein Abtastintervall Δt folgendermaßen (Crassidis, 2006)[17]:

$$v_k = \frac{1}{\Delta t} \int_{t_{k-1}}^{t_k} V(t)\,\mathrm{dt}. \tag{2.177}$$

Die Varianz σ_v^2 von v_k errechnet sich aus der Varianz des zeitkontinuierlichen Rauschens σ_V^2 ($\equiv$ spektrale Leistungsdichte des Systemrauschens) in Abhängigkeit der Abtastzeit Δt:

$$\sigma_v^2 = \frac{\sigma_V^2}{\Delta t}. \tag{2.178}$$

Daraus wird ersichtlich, dass es praktischer ist, die Rauscheigenschaften im Zeitkontinuierlichen zu modellieren und je nach Abtastintervall die tatsächlich wirksame diskrete Varianz zu errechnen.

Ein weißes Rauschen im Zeitkontinuierlichen ist ein theoretisches Konzept (Gelb, 1974). Der Name *weißes Rauschen* weist darauf hin, dass im Leistungsdichtespektrum alle Frequenzen mit gleicher Leistung vertreten sind. Dies entspricht auch der gängigen Definition von weißem Rauschen. Das bedeutet allerdings, dass die mittlere Leistung über alle Grenzen groß ist. Physikalisch ist das nicht möglich (Gelb, 1974). Dennoch lassen sich Prozesse, die in einem bestimmten (endlichen) Frequenzbereich die genannten Eigenschaften aufweisen, sehr gut auf diese Art beschreiben (Hänsler, 2013).

Beispielsweise kann, ausgehend von Gleichung 2.178, das Rauschverhalten von einem Gyroskop in einer numerischen Simulation errechnet werden. Für jeden Simulationsschritt (t) muss ein diskreter normalverteilter Rauschterm $v_\omega(t)$ berechnet werden.

$$v_\omega(t) = \sigma_{v,\omega} \cdot \mu(t). \tag{2.179}$$

Die Funktion $\mu(t)$ liefert eine normalverteilte Zufallszahl (mit $1\,\sigma = 1$, Mittelwert $= 0$):

$$\mu \sim \mathcal{N}(0,1). \tag{2.180}$$

Durch Einfügen von Gleichung 2.178 wird die diskrete Gyroskop Standardabweichung $\sigma_{v,\omega}$ durch die Wurzel der spektralen Leistungsdichte $\sigma_{V,\omega}$ ersetzt:

$$v_\omega(t) = \frac{\sigma_{V,\omega}}{\sqrt{\Delta t}} \cdot \mu(t). \tag{2.181}$$

Die Simulation muss bei dieser Darstellung die Wurzel der spektralen Leistungsdichte ($\sigma_{V,\omega}$) bereithalten und je nach gewünschter Frequenz den passenden Wert für Δt einsetzen. Diese Zusammenhänge werden in Abschnitt 6.3 genutzt, um Sensordaten zu simulieren.

[17] Laut J. L. Crassidis basiert dieses Modell auf Aufzeichnungen von F. Landis Markley (NASA Goddard Space Flight Center).

Üblicherweise beschreiben die Sensorhersteller das Rauschverhalten direkt mit der spektralen Leistungsdichte in den zugehörigen Datenblättern. Alternativ kann die spektrale Leistungsdichte selbst ermittelt werden, siehe dazu Abschnitt 2.10.

Die entsprechenden Einheiten sind in den folgenden Kapiteln für die stochastische Modellierung wichtig. Wenn z. B. wie oben mit Gyroskop Messungen gerechnet wird, liegt für $v_\omega(t)$ die Einheit rad/s vor (alternativ °/s). Somit hat $\sigma_{V,\omega}$ folgende Einheit:

$$\text{rad/s} \cdot \sqrt{s}.$$

Häufig wird $\sqrt{s}$ durch $1/\sqrt{\text{Hz}}$ ausgedrückt:

$$\frac{\text{rad/s}}{\sqrt{\text{Hz}}}. \tag{2.182}$$

Für ein Beschleunigungsmesser wäre die Einheit der Wurzel der spektralen Leistungsdichte:

$$\frac{\text{m/s}^2}{\sqrt{\text{Hz}}}.$$

Eine weitere nützliche Eigenschaft dieser Darstellung ist, dass man auf den Fehler schließen kann, der durch die Integration des normalverteilten Rauschens entsteht (ungeachtet der übrigen Fehlereinflüsse). Ein Winkelfehler aus den diskreten Drehraten (Gl. 2.181) entsteht durch:

$$\Delta\phi = v_\omega \cdot \Delta t. \tag{2.183}$$

Somit ergibt sich die dazugehörige Standardabweichung:

$$\sigma_{\Delta\phi} = \sigma_{v,\omega} \cdot \Delta t = \frac{\sigma_{V,\omega}}{\sqrt{\Delta t}} \Delta t \tag{2.184}$$

$$= \sigma_{V,\omega} \cdot \sqrt{\Delta t}. \tag{2.185}$$

Die Einheit von $\sigma_{\Delta\phi}$ ist folglich (siehe Gl. 2.182):

$$\frac{\text{rad/s}}{\sqrt{\text{Hz}}} \cdot \sqrt{s} = \text{rad}. \tag{2.186}$$

Alle Parameterschätzmodelle in den folgenden Kapiteln, die im Zeitkontinuierlichen modelliert werden, verwenden die spektrale Leistungsdichte um den Fehlereinfluss von weißem Rauschen stochastisch zu berücksichtigen. Bei Gyroskopen spricht man oft von einem *angle random walk*, bei Beschleunigungsmessern von einem *velocity random walk*.

Für das Dreiachs-Gyroskop in dem MEMS Sensor *MPU-9150* gibt der Hersteller InvenSense eine *Noise Density* von 0,005 $\frac{^\circ/\mathrm{s}}{\sqrt{\mathrm{Hz}}}$ an. Somit wäre der durch Rauschen verursachte Fehler nach 100 Sekunden:

$$\sigma_{\Delta\phi} = 0{,}005\,\frac{^\circ/\mathrm{s}}{\sqrt{\mathrm{Hz}}} \cdot \sqrt{100\,\mathrm{s}} = 0{,}05^\circ.$$

Ein weiterer signifikanter Fehler in Trägheitssensoren ist der Biasfehler und sein Änderungsverhalten. Dabei liegt der Fokus auf dem Änderungsverhalten (*Biasdrift*) und weniger auf dem tatsächlichen absoluten Fehler. Idealerweise würde sich der Biasfehler nicht ändern und könnte so durch eine gewissenhafte Vorkalibrierung beseitigt werden. Tatsächlich ändert sich der Biasfehler u. a. bei Temperaturänderungen, Orientierungsänderungen (g-abhängiger Fehler) oder Vibrationen (siehe Abschnitt 4.6).

Die Grundlage für die stochastische Modellierung der Änderung der Biasfehler ist ein Wiener-Prozess bzw. ein Gaußscher Random Walk (Schilling und Partzsch, 2014).

$$b(N \cdot \Delta t) = \sqrt{\Delta t} \cdot \sum_{i=1}^{N} \sigma_b \cdot \mu_i \tag{2.187}$$

mit

$$\mu_i \sim \mathcal{N}(0, 1). \tag{2.188}$$

Oder anders ausgedrückt:

$$b_{i+1} = b_i + \sigma_b \cdot \mu_i \cdot \sqrt{\Delta t}. \tag{2.189}$$

Eine Umsetzung von Gleichung 2.189 als Simulation (siehe Abschnitt 6.3) ist in Abbildung 2.9 zu sehen. Die Simulation zeigt in diesem Beispiel das *Random Walk* Verhalten bei einem Bias Random Walk von $\sigma_b = 0{,}0001\,\frac{\mathrm{rad/s}}{\sqrt{\mathrm{s}}}$. Zum Vergleich zeigt Abbildung 2.10 tatsächlich gemessene Drehraten von einem MEMS Gyroskop der Firma InvenSense in Ruhelage und bei gleichbleibender Temperatur. Die Drehrate der Erde ist in diesen Messungen enthalten, liegt aber in der Größenordnung $10^{-5}\,\mathrm{rad/s}$, sodass die abgebildeten Drehraten als Offsetfehler anzusehen sind. Als Ursache für den Bias Random Walk (BRW) in MEMS Sensoren gibt Woodman (2007) das sogenannte Funkelrauschen bzw. Flackerrauschen (*engl. flicker noise*) an. Diese zufälligen Fehler entstehen in der Sensorelektronik und den angeschlossenen Komponenten. Die Einheit von einem Bias Random Walk lässt sich aus Gleichung 2.189 ableiten. Da der Bias selbst die Einheit des Sensors beibehalten muss, ergibt sich als Einheit (bei der Angabe der Zeit in Sekunden):

$$\frac{\text{Grundeinheit}}{\sqrt{\mathrm{s}}}. \tag{2.190}$$

Für Gyroskope also z. B. $\frac{\mathrm{rad/s}}{\sqrt{\mathrm{s}}}$ und für Beschleunigungsmesser $\frac{\mathrm{m/s^2}}{\sqrt{\mathrm{s}}}$.

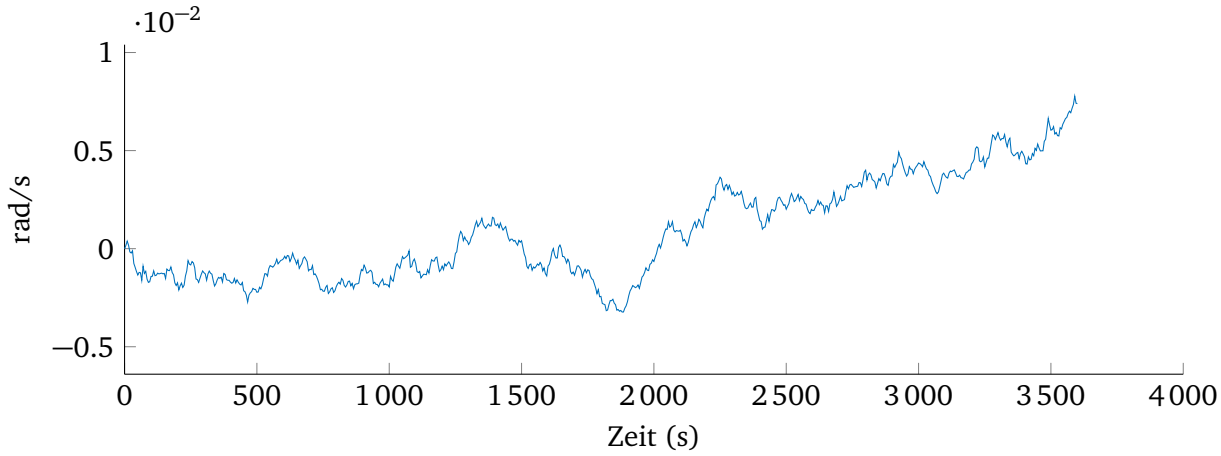


Abbildung 2.9.: Gaußscher Random Walk (simuliert mit Gl. 2.189).

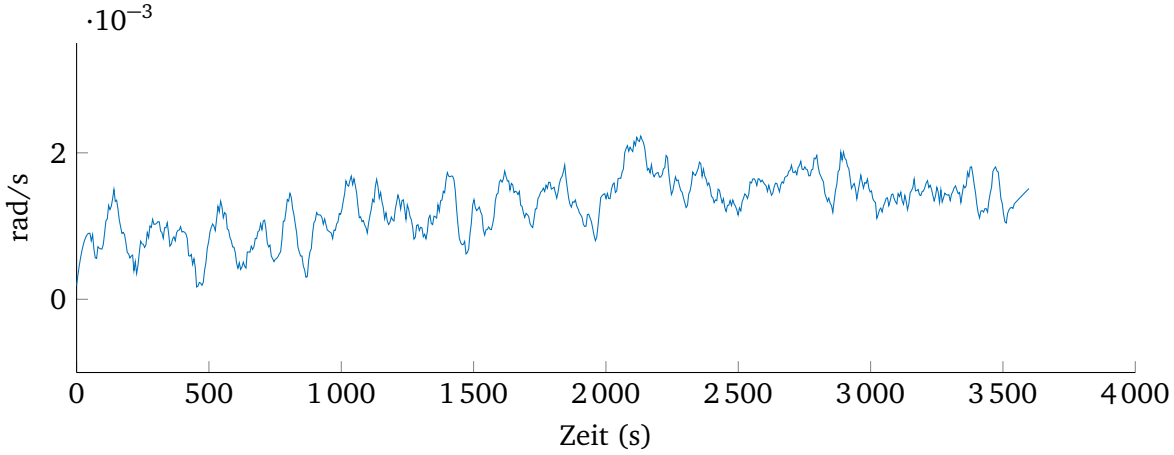


Abbildung 2.10.: Drehraten im Stillstand von InvenSense MPU9150 Gyroskop.

3 Multisensor Navigation

3.1 Grundlagen

Die verwendeten Koordinatensysteme (Tab. 3.1) folgen den in der Geodäsie sowie Luft- und Raumfahrt üblichen Konventionen (Jekeli, 2001, Kap. 1), (Hofmann-Wellenhof et al., 2003, Kap. 3.1), (Farrell, 2008, Kap. 2.2) u. (Wendel, 2011, Kap. 3.1). Der Plattform- sowie der Sensor-Frame werden in Abschnitt 4.3.1 definiert.

Tabelle 3.1.: Koordinatensysteme.

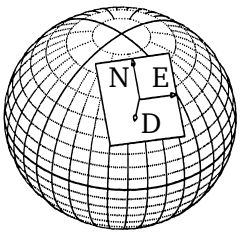
Beschreibung	Englische Bezeichnung	Kürzel
Inertialsystem (Abb. 3.1b)	Earth Centered Inertial (ECI)	i-Frame
Erdfestes System (Abb. 3.1b)	Earth Centered, Earth Fixed (ECEF)	e-Frame
Navigationsframe (Abb. 3.1a)	Navigation Frame (NED)	n-Frame
LAV	Local Astronomical System (LAV)	l-Frame
Körperfestes System	Body Frame at Center of Mass	b-Frame
Plattform System	Platform Frame	p-Frame
Sensor System	Sensor Frame	s-Frame

Das *Inertialsystem* ECI (i-Frame) ist praktisch frei von Beschleunigungen und im Zentrum der Erde gelagert. Inertialsensoren messen bezüglich dieses Koordinatensystems. Im gleichen Ursprung ist das *erdfeste System* (e-Frame) gelagert, folgt aber der Drehung ω der Erde, siehe Abbildung 3.1b. Die Transformation zwischen beiden Systemen beinhaltet zusätzlich noch die Polbewegung P, sowie die Nutation N und Präzession R (Xu und Xu, 2016):

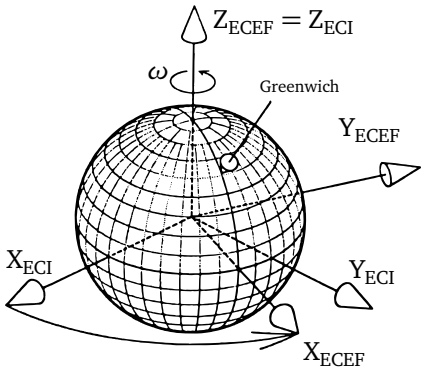
$$\mathbf{R}_i^e(t) = \mathbf{R}_P \cdot \mathbf{R}_\omega \cdot \mathbf{R}_N \cdot \mathbf{R}_R. \tag{3.1}$$

Der *Navigation Frame* (n-Frame) folgt dem zu navigierenden Objekt, die Vertikalachse ist in Richtung der Erde gerichtet und parallel zur Ellipsoidnormalen, siehe Abbildung 3.1c und 3.1a. Bei dem LAV System ist die Vertikalachse parallel zur Schwerebeschleunigung $\mathbf{g}$ bzw. dem Gradienten des Schwerefelds der Erde ∇W ausgerichtet. Die Abweichung zwischen dem n-Frame und dem LAV-Frame lässt sich durch die Lotabweichungen ξ (Nord-Süd) und η (West-Ost) beschreiben, siehe Abbildung 3.1c. Das körperfeste Koordinatensystem (b-Frame) hat seinen Ursprung im Massenschwerpunkt (Center of Mass) des Objekts. Folgende Konventionen werden verwendet (Britting, 1971):

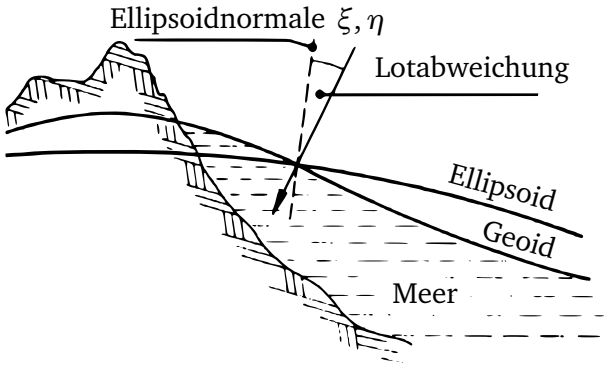
- Rotationsmatrix von System q nach System z: $\mathbf{R}_q^z$
- Position in Koordinatensystem n: $\mathbf{x}^n$



(a) n-Frame (NED). Abbildung: Grewal et al. (2007).



(b) ECEF und ECI. Abbildung: Grewal et al. (2007).



(c) Geoid u. Ellipsoid. Abbildung: Seeber (1988).

Abbildung 3.1.: Koordinatensysteme und Bezugsflächen.

- Drehratenvektor: $\boldsymbol{\omega}_{bc}^{a}$ (c gegenüber b, abgebildet in a)
- Spezifische Kraft: $\mathbf{f}_{ib}^{n}$ Körper b gegenüber Inertialsystem i, abgebildet in n-Frame

Zwischen dem Navigation Frame (n-Frame) und dem erdfesten System (e-Frame) kann transformiert werden, wenn die geographischer Breite B und Länge L des n-Frame Ursprungs bekannt sind (Seeber, 1988):

$$\mathbf{R}_{n}^{e} = \begin{pmatrix} -\sin(B)\cdot\cos(L) & -\sin(L) & -\cos(B)\cdot\cos(L) \\ -\sin(B)\cdot\sin(L) & \cos(L) & -\cos(B)\cdot\sin(L) \\ \cos(B) & 0 & -\sin(B) \end{pmatrix}. \tag{3.2}$$

Der n-Frame ist demnach entlang der Ellipsoidnormalen ausgerichtet, der LAV-Frame ist an der Geoidnormalen ausgerichtet, siehe Abbildung 3.1c. Da der n-Frame der aktuellen Position folgt, ergibt sich in diesem Koordinatensystem eine scheinbare Drehrate, die sogenannte Transportrate. Die Drehrate in rad/s des n-Frames gegenüber dem erdfesten System e ist abhängig von der Geschwindigkeit $\dot{\mathbf{x}}_{eb}^{n}$ (Wendel, 2011, Kap. 3.3):

$$\boldsymbol{\omega}_{en}^{n} = \left(\frac{1}{R_e+h}\cdot\dot{\mathbf{x}}_{eb,y}^{n} \quad -\frac{1}{R_n+h}\cdot\dot{\mathbf{x}}_{eb,x}^{n} \quad -\frac{1}{R_e+h}\cdot\dot{\mathbf{x}}_{eb,y}^{n}\cdot\tan(B)\right)^{T}. \tag{3.3}$$

Bei der Transportrate wird durch den Nord-Süd Krümmungsradius (R_n) und den Ost-West Krümmungsradius (R_e) inkl. der ellipsoidischen Höhe h dividiert. Bei geringer Geschwindigkeit $\dot{\mathbf{x}}^n$ relativ zur Erde kann dieser Korrekturterm vernachlässigt werden.

Die Drehrate zwischen dem Inertialsystem und dem erdfesten System beträgt $\omega = 7,292 \times 10^{-5}$ rad/s und lässt sich bei bekanntem Breitengrad B in den n-Frame umrechnen:

$$\boldsymbol{\omega}_{ie}^{n} = \omega \cdot \begin{pmatrix} \cos(B) \\ 0 \\ -\sin(B) \end{pmatrix}. \tag{3.4}$$

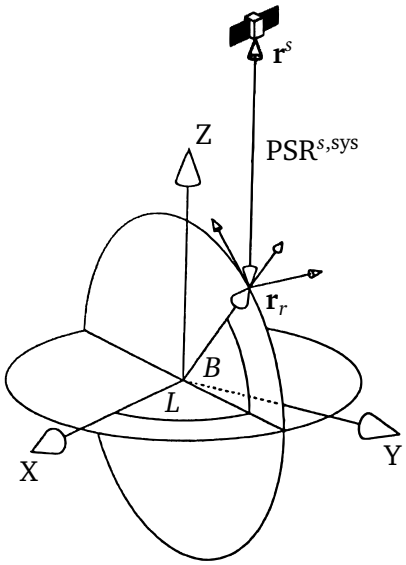


Abbildung 3.2.: GNSS Pseudorangemessung $\mathrm{PSR}^{s,\mathrm{sys}}$. Abbildung: Grewal et al. (2007).

3.2 Überblick

Unter *Navigation* wird die Bestimmung von Position, Geschwindigkeit und Lage verstanden (Wendel, 2011)[1]. D. h. gesucht wird der Navigationszustandsvektor $\mathbf{y}$, beispielsweise im erdfesten Koordinatensystem [2] ECEF (e):

$$\mathbf{y} = \begin{pmatrix} \mathbf{x}^e & \dot{\mathbf{x}}^e & \ddot{\mathbf{x}}^e & \mathbf{q}_b^e & \boldsymbol{\omega}_{eb}^b & \dot{\boldsymbol{\omega}}_{eb}^b \end{pmatrix}^T. \tag{3.5}$$

Bestehend aus den 19 Parametern: der Position $\mathbf{x}^e$, der Geschwindigkeit relativ zur Erde $\dot{\mathbf{x}}^e$ (engl. *ground speed*) und dem Lagequaternion $\mathbf{q}_b^e$, das die Orientierung des zu navigierenden Körpers relativ zur Erde beschreibt. Sowie weiteren (besonders für die Luftfahrt wichtigen) Größen: der Beschleunigung des Kör-

[1] Das Vorgeben einer Route wird dagegen *Guidance* genannt.

[2] In Kapitel 4 wird der Navigationszustandsvektor im n-Frame parametrisiert. Die Wahl des Koordinatensystems kann als Designentscheidung verstanden werden und hängt auch z. T. von den verwendeten Sensoren ab.

pers $\ddot{\mathbf{x}}^e$, den Drehraten relativ zur Erde $\boldsymbol{\omega}_{eb}^b$ und den Winkelbeschleunigungen $\dot{\boldsymbol{\omega}}_{eb}^b$. Das Lagequaternion $\mathbf{q}$ besteht aus einem Realteil q_0, sowie den Imaginärteilen q_1, q_2, q_3 (Kuipers, 1999):

$$\mathbf{q} = q_0 + q_1 \cdot i + q_2 \cdot j + q_3 \cdot k \tag{3.6}$$

mit

$$i^2 = j^2 = k^2 = i \cdot j \cdot k = -1. \tag{3.7}$$

Bzw. in Vektordarstellung:

$$\mathbf{q} = \begin{pmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{pmatrix}. \tag{3.8}$$

Das Lagequaternion kann jederzeit in eine Drehmatrix umgerechnet werden (Jekeli, 2001):

$$\mathbf{R}_b^e\left(\mathbf{q}_b^e\right) = \begin{pmatrix} 1-2\cdot(q_2^2+q_3^2) & -2\cdot q_0\cdot q_3 + 2\cdot q_1\cdot q_2 & 2\cdot q_0\cdot q_2 + 2\cdot q_1\cdot q_3 \\ 2\cdot q_0\cdot q_3 + 2\cdot q_1\cdot q_2 & 1-2\cdot(q_1^2+q_3^2) & -2\cdot q_0\cdot q_1 + 2\cdot q_2\cdot q_3 \\ -2\cdot q_0\cdot q_2 + 2\cdot q_1\cdot q_3 & 2\cdot q_0\cdot q_1 + 2\cdot q_2\cdot q_3 & 1-2\cdot(q_1^2+q_2^2) \end{pmatrix} \tag{3.9}$$

Die Inertiale Navigation erlaubt die Bestimmung dieser Größen und hat eine Sonderstellung, da sie keine Abhängigkeit zu externen Systemen oder Merkmalen der Umwelt[3] hat. Bei bekannter Anfangsposition, Geschwindigkeit und Ausrichtung wird rein durch das Erfassen der Winkelgeschwindigkeit und der Beschleunigung (*specific force*) fortlaufend der Navigationszustand nach den Newtonschen Gesetzen mitgeführt. Farrell (2008) nennt den Zusammenhang in Gl. 3.10 zwischen der spezifischen Kraft $\mathbf{f}^i$ und der tatsächlichen Körperbeschleunigung $\ddot{\mathbf{x}}$ die *Fundamentalgleichung der Inertialen Navigation*.

$$\mathbf{f}^i = \ddot{\mathbf{x}}^i - \mathbf{g}^i \tag{3.10}$$

Eine externe Störung der Inertialen Navigation ist in Sonderfällen zwar möglich, z. B. durch akustische Anregung über kurze Distanzen von Sensorstrukturen bestimmter Hersteller, siehe Trippel et al. (2017) oder Tu et al. (2018), aber dennoch um ein vielfaches schwieriger als bei anderen Navigationssystemen, sodass man gewissermaßen von einem störungssicheren System sprechen kann.

Da verschiedene Navigationsverfahren unterschiedliche Eigenschaften haben, spricht man von *Integrierter Navigation*, wenn verschiedene Messverfahren kombiniert werden. Das wichtigste Beispiel ist hierfür die Fusion von *Inertialer Navigation* bzw. einem *Inertial Navigation Systems* (INS) basierend auf

[3] Vorwissen über den Planeten zur Kompensation der Schwerebeschleunigung und Planetendrehrate ist aber dennoch notwendig.

den Daten von *Inertial Measurement Units* (IMUs) und einem oder mehreren *Satellitennavigationssystemen* (GNSS). Das INS kann ausgehend von einer bekannten Anfangsnavigationslösung fortlaufend mit hoher Datenrate, sehr niedriger Latenz und ständiger Verfügbarkeit eine vollständige Navigationslösung bereitstellen. Der Nachteil von einem reinen INS ist dafür, dass die Navigationslösung nur kurzzeitig genau ist und mit der Zeit immer weiter divergiert. GNSS dagegen ist langzeitstabil, stellt aber mit nur einer Antenne[4] keine Lageinformation zur Verfügung. Auch die Sichtbarkeit zu den Satelliten ist nicht garantiert. Hinzu kommt, dass die GNSS Messfrequenz im Vergleich zu einem INS gering ist. Zudem weisen GNSS Messungen i. d. R. eine hohe Latenz auf. Die Vereinigung beider Systeme liegt nahe. Dieses Kapitel geht zuerst auf die für Kapitel 5 relevanten Aspekte der IMU Datenverarbeitung ein. Kleine Fehler addieren sich hier schnell zu signifikanten Fehlern, daher wird dieses Thema tiefer beleuchtet. Tabelle 3.2 zeigt, wie Inertiale Navigationssyteme eingestuft werden können (VectorNav, 2016). Die

Tabelle 3.2.: INS Güteklassen.

Klasse	Kosten	Positionsfehler
Marine Grade	> $ 1 000 000	1800 m pro Tag
Navigation Grade	ca. $ 1 00 000	1500 m pro Stunde
Automotive & Consumer Grade	< $ 100	Nur mit Stützinformationen sinnvoll

Nutzung von Sensoren auf Automotive und Consumer Grade Basis (Low-Cost MEMS) stellt hohe Anforderungen an die Algorithmen, da verschiedene Fehlereinflüsse kompensiert und zum Teil fortlaufend kalibriert werden müssen. Abschnitt 3.4 geht auf die verschiedenen Fehlereinflüsse von IMUs im Detail ein. Die Anfangsorientierung bzw. das *Initial Alignment* wird üblicherweise aus den IMU Beobachtungen gewonnen. Für Low-Cost IMU muss für die Azimutbestimmung auf Magnetometer zurückgegriffen werden. Die Winkel ϕ (Roll) und θ (Pitch), bezogen auf den n-Frame, können im Stillstand aus der von einem kalibrierten Accelerometer gemessenen spezifischen Kraft $\mathbf{f}^b$ geschätzt werden[5]

$$\mathbf{f}^b = \begin{pmatrix} f_x^b \\ f_y^b \\ f_z^b \end{pmatrix}. \tag{3.11}$$

[4] Kis und Lantos (2014) zeigen, dass mit drei Low-Cost GNSS Empfängern (u-blox 6T) sowie einer Low-Cost IMU und kurzen Antennenabständen eine zuverlässige Lösung der Phasenmehrdeutigkeiten möglich ist, was dann für die Attitudeschätzung genutzt werden kann. Roth et al. (2012) zeigen, dass mit nur zwei GNSS Low-Cost Empfängern und sehr kurzen Antennenabständen sowie einer Low-Cost IMU eine Azimutbestimmung möglich ist. Beide Ansätze basieren im Kern auf der LAMBDA Methode (Teunissen, 1995).

[5] Idealerweise wird die gemessene spezifische Kraft über einen längeren Zeitraum gemittelt und um Ausreißer bereinigt (z. B. über ein Medianfilter). Zudem sollte auch ein tatsächlicher Stillstand in den Beobachtungen erkannt werden, siehe dazu Abschnitt 3.8.

Da die spezifische Kraft im Stillstand der Schwerebeschleunigung mit umgekehrtem Vorzeichen entspricht (Farrell, 2008, Kap. 10.3):

$$\hat{\phi}_b^n = \arctan2\left(-f_y^b, -f_z^b\right) \tag{3.12}$$

$$\hat{\theta}_b^n = \arctan2\left(f_x^b, \sqrt{(f_y^b)^2 + (f_z^b)^2}\right). \tag{3.13}$$

Eine Genauigkeitsabschätzung auf Basis der Gaußschen Fehlerfortpflanzung dazu findet sich in (Schüler, 1997, S. 47) und sollte bei der Kalman-Filter Initialisierung in der Kovarianzmatrix $\mathbf{Q}_{yy}$ berücksichtigt werden, um das Initial Alignment stochastisch richtig zu beschreiben. Das Azimut lässt sich mit kalibrierten Gyroskopen im Stillstand bestimmen, wenn die Beobachtungen genau genug sind um die Erddrehrate von 15,0408°/h ($7,292 \times 10^{-5}$ rad/s) zu erfassen und eine entsprechende Bias-Stabilität haben[6]; man spricht auch vom *gyro-compassing*. Je nach Rauschverhalten muss dazu eine mehr oder weniger lange Zeitreihe an Beobachtungen erfasst werden. Mit den in die lokale Ebene n projizierten und gemittelten Winkelgeschwindigkeiten lässt sich das Azimut berechnen (Wendel, 2011, Kap. 3.5):

$$\hat{\psi}^n = \arctan2\left(-\omega_{ib,y}^n, \omega_{ib,x}^n\right). \tag{3.14}$$

Analog dazu lässt sich die kalibrierte, gemittelte und in die Ebene n projizierte Messung $\mathbf{m}^n$ von einem Magnetometer nutzen um geographisch Nord zu bestimmen (Wendel, 2011, Kap. 10.1):

$$\hat{\psi}^n = d - \arctan2\left(m_y^n, m_x^n\right). \tag{3.15}$$

Für die Projektion in die Ebene werden ϕ und θ benötigt. Daraus lässt sich eine reduzierte Transformationsmatrix $\bar{\mathbf{R}}_b^n$ bilden:

$$\bar{\mathbf{R}}_b^n = \begin{pmatrix} \cos(\theta) & \sin(\phi)\cdot\sin(\theta) & \sin(\theta)\cdot\cos(\phi) \\ 0 & \cos(\phi) & -\sin(\phi) \\ -\sin(\theta) & \cos(\theta)\cdot\sin(\phi) & \cos(\theta)\cdot\cos(\phi) \end{pmatrix}. \tag{3.16}$$

Geographisch Nord muss dazu natürlich ebenfalls definiert und für die Anwendung gewählt sein, beispielsweise durch den International Terrestrial Reference Frame (ITRF). Der Deklinationskorrekturterm d lässt als Funktion der geographischen Position und der aktuellen Zeit berechnen: $d = f(B, L, h, t)$. Üblicherweise wird d über Kugelfunktionskoeffizienten modelliert, so wie z. B. im World Magnetic Model (Chulliat et al., 2015). Alternativ lässt sich daraus auch ein Gittermodell zur schnellen Interpolation der Deklination vorausberechnen (Zwiener, 2012). Der Korrekturterm d aus dem WMM 2015 hat in Mitteleuropa eine Genauigkeit von ca. $0,3° - 0,8°$ (Chulliat et al., 2015). Diese Unsicherheit muss ebenfalls bei der Initialisierung der Zustandsschätzung berücksichtigt werden.

[6] Dies schließt nach aktuellem Stand der Technik die meisten MEMS Gyroskope aus.

3.3 Strapdown Rechnung

Es gibt zwei Arten von IMUs: a) kardanisch aufgehängte und mechanisch im Raum stabilisierte[7] IMUs und b) fest mit der Plattform verbundene IMUs. Es werden im Folgenden nur die kostengünstigen, kompakten und mechanisch weniger komplexen Strapdown IMUs betrachtet. Während die mechanische Komplexität hier reduziert ist, vergrößert sich der algorithmische Aufwand. Die hochfrequenten Sensorsignale müssen fortlaufend aufintegriert werden um die Plattformausrichtung, Geschwindigkeit und Position im Raum nicht zu verlieren. Diese Berechnungen nennt man *Strapdown Algorithmus*. Abbildung 3.3 stellt das Ablaufprinzip dar.

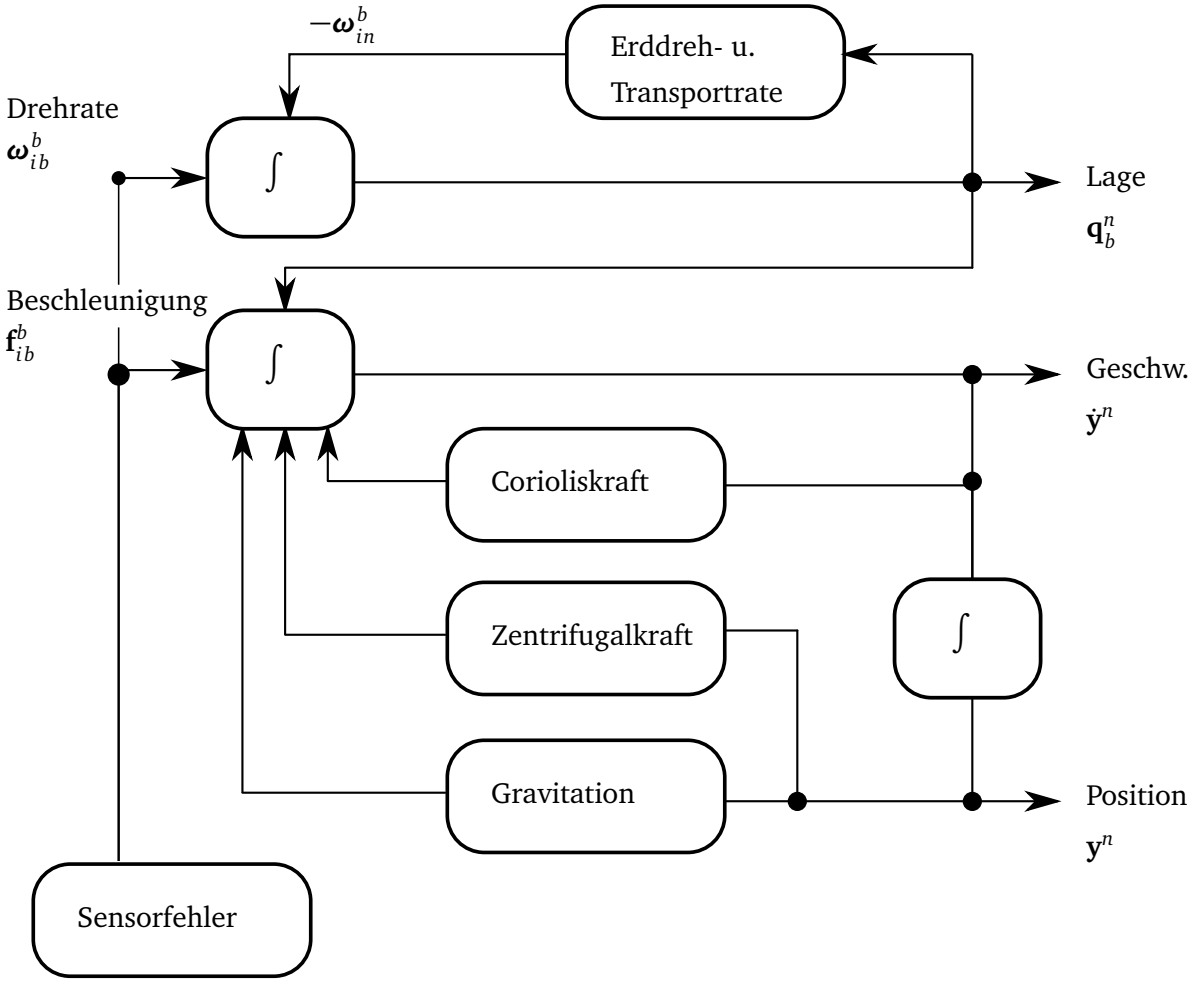


Abbildung 3.3.: Blockschaltbild Strapdown Algorithmus.

[7] Motoren an der Aufhängung regeln abhängig von den gemessenen Drehraten ein Drehmoment, sodass die Drehraten bei 0 °/s bleiben. Dadurch behalten die Beschleunigungsmesser ihre Orientierung im Startkoordinatensystem und können aufintegriert werden.

Im Folgenden werden die dazu benötigten Differentialgleichungen betrachtet. Angefangen bei der Orientierung, das zeitliche Verhalten von Quaternionen in Abhängigkeit der Drehraten ist:

$$\dot{\mathbf{q}}_b^n = \frac{1}{2}\mathbf{q}_b^n \cdot \begin{pmatrix} 0 \\ \boldsymbol{\omega}_{nb}^b \end{pmatrix}. \tag{3.17}$$

Die Winkelgeschwindigkeit $\boldsymbol{\omega}_{nb}^b$ lässt sich nach einer Korrektur der Erddrehrate $\boldsymbol{\omega}_{ie}^n$ sowie der Transportrate des n-Frames aus den Gyroskopmessungen $\boldsymbol{\omega}_{ib}^b$ errechnen (Wendel, 2011, Kap. 3.3):

$$\boldsymbol{\omega}_{nb}^b = \boldsymbol{\omega}_{ib}^b - \mathbf{R}_n^b \cdot \left(\boldsymbol{\omega}_{ie}^n - \boldsymbol{\omega}_{en}^n\right). \tag{3.18}$$

Bei geringen Geschwindigkeiten kann die Transportrate ignoriert werden. Auch die Erddrehrate (15°/h) kann bei geringen Genauigkeitsanforderungen ignoriert werden (z. B. bei Spielzeug UAVs). Beide Größen hängen von der geographische Breite B ab, siehe (Wendel, 2011, Kap. 3.3). Wenn diese nicht bekannt ist, muss sogar gezwungenermaßen auf die Korrektur verzichtet werden.

Die Differentialgleichung 3.17 lässt sich in einer geschlossenen Form[8] als i. A. nicht kommutatives Produkt von zwei Quaternionen (Operator $\bullet$) über den Zeitraum Δt hinweg aufintegrieren (Jekeli, 2001, S. 114-117), (Farrell, 2008, Kap. D.5), (Wendel, 2011, Kap. 3.3), (Zwiener, 2012, Kap. 3.3):

$$\mathbf{q}_b^n(t+\Delta t) = \mathbf{q}_b^n(t) \bullet \begin{pmatrix} \cos(\frac{1}{2} \cdot |\boldsymbol{\omega}_{nb}^b| \cdot \Delta t) \\ \frac{1}{|\boldsymbol{\omega}_{nb}^b \cdot \Delta t|} \cdot \boldsymbol{\omega}_{nb}^b \cdot \Delta t \cdot \sin(\frac{1}{2} \cdot |\boldsymbol{\omega}_{nb}^b| \cdot \Delta t) \end{pmatrix}. \tag{3.19}$$

Das Integrator-Quaternion wird durch einen Kosinus Term im skalaren Realteil gebildet, die drei Komponenten des Imaginärteils werden über den Sinus Term und dem normalisierten Drehratenvektor berechnet. Dies enthält die Annahme, dass die Drehraten in der Zeit Δt konstant bleiben. Falls diese Annahme nicht zutrifft, kann dies aufgrund der Nichtkommutativität von Drehbewegungen letztendlich nur durch kleinere Zeitschritte Δt, also höheren IMU Abtastraten ausgeglichen werden.

Die Bewegungsgleichung ist im inertialen System einfach aufzubauen:

$$\ddot{\mathbf{x}}^i = \mathbf{f}^i + \mathbf{g}^i. \tag{3.20}$$

Die tatsächliche Objektbeschleunigung setzt sich aus der spezifischen Kraft $\mathbf{f}_{ib}^i$ und der Schwerebeschleunigung $\mathbf{g}^i$ im Inertialsystem zusammen. Für die folgenden Kapitel wurde aber als Bezugssystem der Navigation-Frame (n-Frame) gewählt, da die Aufteilung in Lage und Höhe gut lesbar sowie interpretierbar ist und z. B. bei der Einbindung von Höhenmessungen sehr einfache Beobachtungsgleichungen resultieren. Da es sich um ein drehendes System handelt, müssen zusätzlich zur Massenanziehung noch

[8] Gl. 3.19 ist nicht für den Sonderfall definiert, dass der Betrag der Drehraten gleich 0 ist. Um diese Division durch 0 zu umgehen, schlägt Wendel (2011) eine Reihenentwicklung vor bzw. in diesem Fall ist auch keine Änderung notwendig.

Scheinbeschleunigungen ($\mathbf{f}^n_{\text{Centrifugal}}$ und $\mathbf{f}^n_{\text{Coriolis}}$) berücksichtigt werden. Die Navigationsgleichung im n-Frame lautet (Hofmann-Wellenhof et al., 2003, Kap. 11.3.3)[9]:

$$\ddot{\mathbf{x}}^n = \mathbf{f}^n_{\text{Gravitation}} - \mathbf{f}^n_{\text{Centrifugal}} + \underbrace{\mathbf{R}^n_b \cdot \mathbf{f}^b}_{\text{spec. force}} \underbrace{-\left(\boldsymbol{\Omega}^n_{in} + \boldsymbol{\Omega}^n_{ie}\right) \cdot \dot{\mathbf{x}}^n}_{\mathbf{f}^n_{\text{Coriolis}}}, \tag{3.21}$$

mit der Kreuzproduktmatrix $\boldsymbol{\Omega}^n_{ie} = [\boldsymbol{\omega}^n_{ie}\times]$ aus der Erddrehrate[10] $\omega = 7,292 \times 10^{-5}$ rad/s im n-Frame, sowie der Transportrate $\boldsymbol{\Omega}^n_{in} = [\boldsymbol{\omega}^n_{in}\times]$ des n-Frames gegenüber dem Inertialsystem (Wendel, 2011, Kap. 3.3):

$$\boldsymbol{\omega}^n_{ie} = \omega \cdot \begin{pmatrix} \cos(B) \\ 0 \\ -\sin(B) \end{pmatrix} \tag{3.22}$$

$$\boldsymbol{\omega}^n_{in} = \begin{pmatrix} \frac{(\dot{\mathbf{x}}^n_{eb})_y}{R_e+h} \\ -\frac{(\dot{\mathbf{x}}^n_{eb})_x}{R_n+h} \\ -\frac{(\dot{\mathbf{x}}^n_{eb})_y \cdot \tan(B)}{R_e+h} \end{pmatrix}. \tag{3.23}$$

Die Specific Force $\mathbf{f}^b$ im Zentrum des körperfesten Koordinatensystems kann aus verschiedenen verteilten Sensoren bestimmt werden. Hier sei auf Abschnitt 4.3 verwiesen. Die Integration von Gl. 3.21 erfolgt numerisch, die entsprechenden Gleichungen werden gesondert in Abschnitt 3.3.1 betrachtet. Bei Gl. 3.21 wird angenommen, dass die Drehrate und die Beschleunigung konstant bleiben, das heißt wiederum, dass bei konstanter Drehrate sich die Richtung der konstanten Beschleunigung über die Integrationsperiode Δt hinweg verändert. Dieser Zusammenhang führt, falls er nicht korrigiert wird, zu signifikanten Positionsfehlern, daher wird dieser Aspekt in Abschnitt 3.3.2 gesondert betrachtet.

Für praktische Umsetzungen in Softwarealgorithmen sei darüber hinaus auf die sehr ausführlichen Veröffentlichungen von Savage (1998a,b) hingewiesen.

3.3.1 Numerische Aspekte

Für die numerische Integration der Sensordaten empfehlen Titterton und Weston (2004) folgende Algorithmen:

- Euler-Vorwärts Integrationsregel
- Trapez Integrationsregel
- Simpson Integrationsregel

[9] Hofmann-Wellenhof et al. (2003) beschreibt weitere mögliche Parametrisierungen der Navigationsgleichungen in verschiedenen Koordinatensystemen inkl. jeweils einer ausführlichen Herleitung.

[10] Die Drehrate der Erde um die Sonne beträgt 0,041 °/h und wird vernachlässigt.

Die verschiedenen Algorithmen wurden bezüglich ihrer numerischen Genauigkeit untersucht. Darüber hinaus wurde der VERLET-Algorithmus[11] untersucht, der bei physikalischen Simulationen (besonders molekulardynamischen Simulationen) aufgrund der numerischen Stabilität beliebt ist (Verlet, 1967), (Press et al., 2007, Kap. 17.4), (Swope et al., 1982). Da dieser Algorithmus sonst in der Literatur im Zusammenhang mit Strapdown Algorithmen nicht erwähnt wird, betrachtet dieser Abschnitt auch seine Eignung eben für diesen Zweck.

Dazu wurde mit der SIMA Software (Jäger et al., 2012, 2013b) eine Trajektorie generiert. Damit sind fehlerfreie IMU Beobachtungen und die wahren Positionen (*Ground Truth*) verfügbar. Somit können direkt die numerischen Eigenschaften der verschiedenen Algorithmen betrachtet werden. Abbildung 3.4 zeigt die simulierte Trajektorie in Helix-Form. Die Helix hat eine konstante Steigung entlang der Vertikalkomponente, dafür aber eine ständige Änderung der Beschleunigungen auf den horizontalen Komponenten.

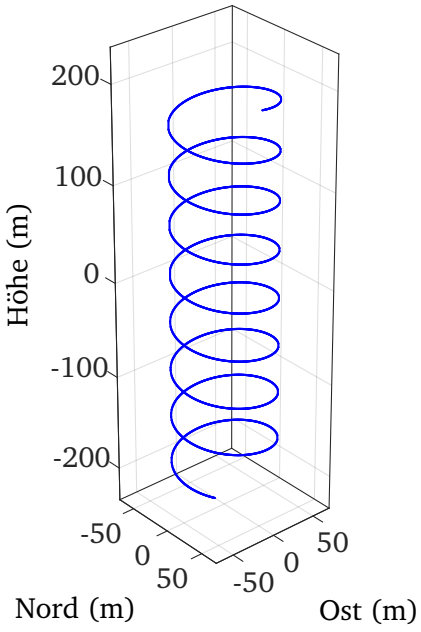


Abbildung 3.4.: Simulierte Helix Trajektorie, generiert von der SIMA Software.

Die Geschwindigkeitsänderung $\Delta\mathbf{v}$ vom Zeitpunkt t zum Zeitpunkt $t+\Delta t$ ist gemäß der Navigationsgleichung 3.21 beispielhaft für den n-Frame (Titterton und Weston, 2004):

$$\dot{\mathbf{x}}^n(t+\Delta t) = \dot{\mathbf{x}}^n(t) + \Delta\mathbf{v}^n. \tag{3.24}$$

[11] Loup Verlet, franz. Physiker.

Der diskrete Geschwindigkeitsänderungsvektor $\Delta\mathbf{v}^n$ wird durch Integration der Beschleunigungen errechnet (ausgehend von Gl. 3.21):

$$\Delta\mathbf{v}^n = \mathbf{R}_b^n(t) \cdot \left(\mathbf{f}^b(t) \cdot \Delta t + \underbrace{\mathbf{A} \cdot \mathbf{f}^b(t) + \mathbf{B} \cdot \mathbf{f}^b(t)}_{\text{Rotationskorrektur}} \right) + \left(\mathbf{f}^n_{\text{Gravitation}}(t) - \mathbf{f}^n_{\text{Coriolis}}(t) - \mathbf{f}^n_{\text{Centrifugal}}(t) \right) \cdot \Delta t \tag{3.25a}$$

$$\Delta\mathbf{v}^n = \ddot{\mathbf{x}}^n \cdot \Delta t + \mathbf{R}_b^n(t) \cdot \left(\mathbf{A} + \mathbf{B} \right) \cdot \mathbf{f}^b(t) \tag{3.25b}$$

Auf die (wichtigen) Rotationskorrekturmatrizen **A** und **B** wird an dieser Stelle nicht weiter eingegangen, dem Thema ist der Abschnitt 3.3.2 gewidmet. Matrix **A** kann mit Gl. 3.39 berechnet werden, Matrix **B** mit Gl. 3.43. Bei den nachfolgenden Untersuchungen ist die Korrektur enthalten. In Gleichung 3.25a ist darauf zu achten, dass die Schwerebeschleunigung (engl. *gravity*) hier in die Beschleunigung durch die Massenanziehung (engl. *gravitation*) und Zentrifugalbeschleunigung aufgeteilt wurde. Falls das Schweremodell nur die gesamte Schwerebeschleunigung liefert, so ist Gl. 3.25a entsprechend anzupassen.

Im folgenden Abschnitt wird auf die Angabe des Bezugskoordinatensystems verzichtet, da die Berechnungsmethoden allgemeingültig sind, solange das $\Delta\mathbf{v}$ aus Gleichung 3.25a aus den passenden Navigationsgleichungen ermittelt wurde.

Euler-Vorwärts: Diese Integrationsmethode wird auch explizites Euler-Verfahren genannt oder im engl. *Euler forward method* und ist ein Verfahren, das einfach ausgehend von der aktuellen Position und Geschwindigkeit die nächste Position berechnet:

$$\dot{\mathbf{x}}(t + \Delta t) = \dot{\mathbf{x}}(t) + \Delta\mathbf{v} \tag{3.26}$$

$$\mathbf{x}(t + \Delta t) = \mathbf{x}(t) + \dot{\mathbf{x}}(t) \cdot \Delta t. \tag{3.27}$$

Aufgrund der Linearisierung profitiert dieses Verfahren am meisten von kurzen Integrationsschrittweiten. Algorithmischer Positionsfehler nach 500 Sek.: 1,015 m, siehe Abbildung 3.5.

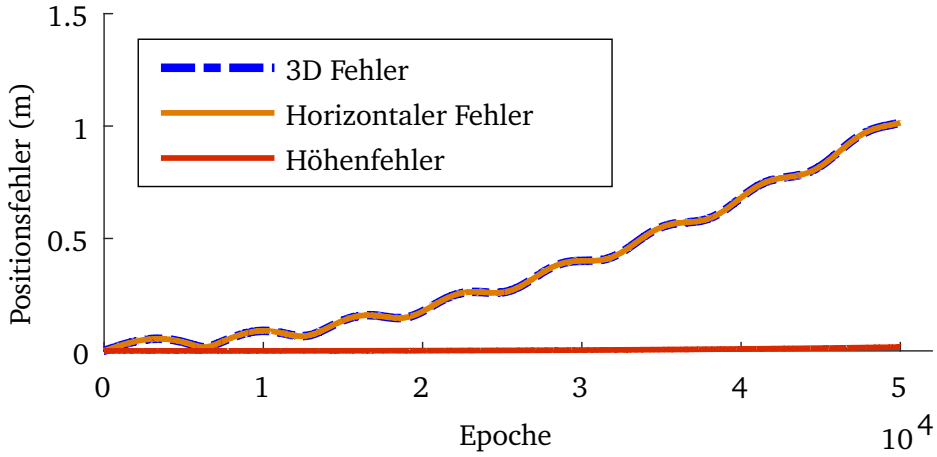


Abbildung 3.5.: Algorithmischer Fehler durch Euler-Vorwärts Integrationsalgorithmus bei dem Tracking der Helix Trajektorie (Abb. 3.4).

Trapez Integration: Bei der Trapez Integration wird die mittlere Geschwindigkeit entlang der Bewegung für die Integration genutzt:

$$\dot{\mathbf{x}}(t+\Delta t) = \dot{\mathbf{x}}(t) + \Delta \mathbf{v} \tag{3.28}$$

$$\mathbf{x}(t+\Delta t) = \mathbf{x}(t) + \frac{1}{2}\Big(\dot{\mathbf{x}}(t) + \dot{\mathbf{x}}(t+\Delta t)\Big) \cdot \Delta t. \tag{3.29}$$

Algorithmischer Positionsfehler nach 500 Sek.: 1,005 m, siehe Abbildung 3.6.

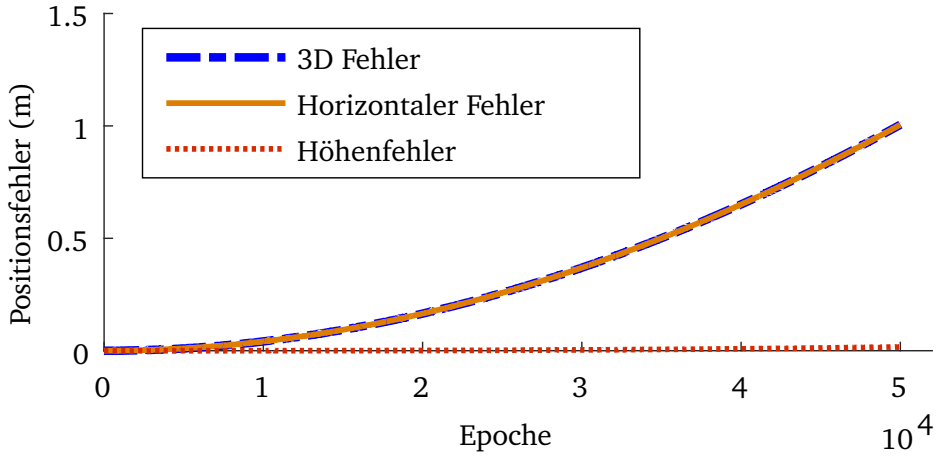


Abbildung 3.6.: Algorithmischer Fehler durch Trapez Integrationsalgorithmus bei dem Tracking der Helix Trajektorie (Abb. 3.4).

Simpson Integration: Bei der Integration nach Simpson[12] wird der Funktionsverlauf über eine Parabel angenähert:

$$\dot{\mathbf{x}}(t+\Delta t) = \dot{\mathbf{x}}(t) + \Delta \mathbf{v} \tag{3.30}$$

$$\mathbf{x}(t+\Delta t) = \mathbf{x}(t) + \frac{1}{6}\Big(\dot{\mathbf{x}}(t-\Delta t) + 4\cdot\dot{\mathbf{x}}(t) + \dot{\mathbf{x}}(t+\Delta t)\Big)\cdot\Delta t. \tag{3.31}$$

Algorithmischer Positionsfehler nach 500 Sek.: 1,015 m, siehe Abbildung 3.7.

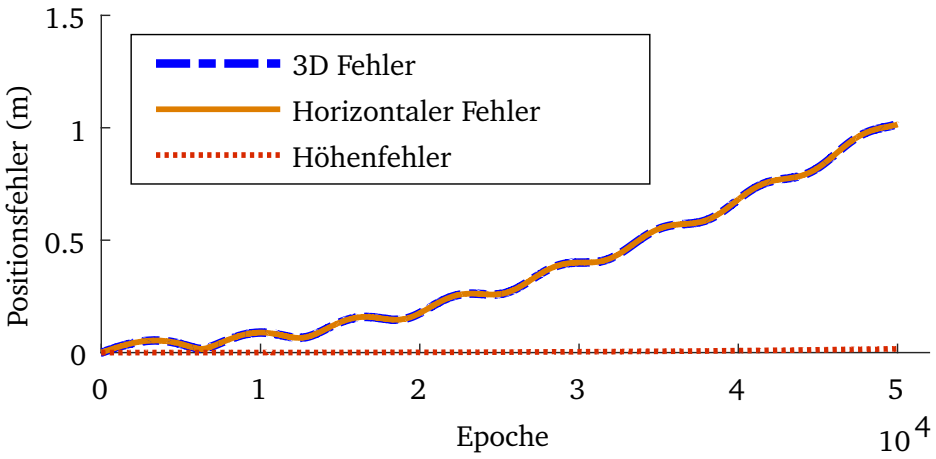


Abbildung 3.7.: Algorithmischer Fehler durch Simpson Integrationsalgorithmus bei dem Tracking der Helix Trajektorie (Abb. 3.4).

Verlet Integration: Die Eignung der Verlet Integration für die Inertialnavigation ist, soweit dem Autor bekannt, nicht beschrieben. Bei der Verlet Methode muss eine Schätzung für die vorhergehende Position $\mathbf{x}(t-\Delta t)$ vorliegen, für eine Diskussion der Methode sei auf (Press et al., 2007, Kap. 17.4) verwiesen. Die Gleichung lautet (Swope et al., 1982):

$$\mathbf{x}(t+\Delta t) = 2\cdot\mathbf{x}(t) - \mathbf{x}(t-\Delta t) + \ddot{\mathbf{x}}(t)\cdot\Delta t^2. \tag{3.32}$$

Algorithmischer Positionsfehler nach 500 Sek.: 1,091 m, siehe Abbildung 3.8.

[12] Nach Thomas Simpson, engl. Mathematiker *1710, †1761.

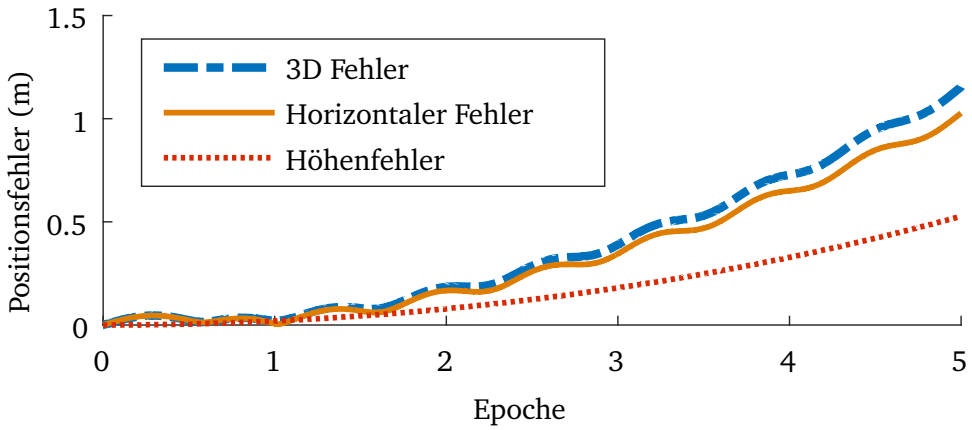


Abbildung 3.8.: Verlet Strapdown Integrationsalgorithmus.

Bei allen Ansätzen (bis auf Verlet) ist der Höhenkanal durch die konstante Steigung der Helix praktisch fehlerfrei integrierbar. Der Verlet Algorithmus kann daher, zumindest für die vorliegende Trajektorie, nicht empfohlen werden.

Nach Savage (1998b) soll der Fehler in einem INS, der durch numerische Integration der Messungen entsteht, weniger als 5 % zum Gesamtfehler beitragen. Ein gutes INS ohne Positionsmessungen zur Stützung hat einen Positionsdrift von ca. 1500 m nach einer Stunde[13]. 5 % davon sind etwa 75 m. Die Unterschiede zwischen den Verfahren sind gering, auch der Rechenaufwand der Methoden spielt bei heutigen Computersystemen keine signifikante Rolle mehr. Die Unterschiede verschwinden noch weiter, wenn die Abtastrate erhöht wird. Bei der untersuchten synthetischen Trajektorie sind die Messungen frei von Fehlern und Rauschen; bei verrauschten Daten und eher gleichmäßigen Bewegungen kann die Simpson Integration durch ihre Mittelwertbildung Vorteile haben. Für das in Kapitel 5 beschriebene System wurde die Trapez Integration verwendet.

3.3.2 Rotationskorrektur

Dieser Abschnitt behandelt die in Gleichung 3.25a auftauchenden Rotationskorrekturen. Gesucht ist die Geschwindigkeitsänderung ($\Delta\mathbf{v}^n$) über den Zeitabschnitt Δt. Bekannt ist die aktuelle Drehrate ($\boldsymbol{\omega}_{nb}^b$) sowie die *Specific Force* ($\mathbf{f}^b$). Es wird eine konstante Drehrate und Beschleunigung über den Zeitraum Δt angenommen. Da sich somit die Drehmatrix $\mathbf{R}_b^n(t)$ folglich konstant ändert, kann die Integration der Beschleunigungen nicht einfach nur über eine Multiplikation mit Δt erfolgen. Die Schwerebeschleunigung sowie die Coriolis-Beschleunigung aus Gleichung 3.25a kann über kurze Zeiträume hinweg als konstant angenommen werden. Der Fokus liegt somit auf folgendem Teil von Gl. 3.25a, gekennzeichnet mit dem *s* Subskript:

$$\Delta\mathbf{v}_s^n = \int_t^{t+\Delta t} \mathbf{R}_b^n(t, \boldsymbol{\omega}_{nb}^b) \cdot \mathbf{f}^b \ \mathrm{dt}. \tag{3.33}$$

[13] Siehe Tabelle 3.2

Die bekannte Orientierung zum Zeitpunkt t wird vor das Integral gezogen, die Drehung bis zum Zeitpunkt $t + \Delta t$ wird über die e-Funktion abgebildet (Wendel, 2011):

$$\Delta \mathbf{v}_s^n = \mathbf{R}_b^n(t) \int_t^{t+\Delta t} e^{\int_t^{t+\Delta t} [\boldsymbol{\omega}_{nb}^b \times] \cdot \mathrm{dt}} \cdot \mathbf{f}^b \ \mathrm{dt}. \tag{3.34}$$

Die strenge Lösung für das Integral dieser e-Funktion lautet (Titterton und Weston, 2004, S. 312):

$$e^{\int_t^{t+\Delta t} \boldsymbol{\Omega}_{nb}^b \ \mathrm{dt}} = \mathbf{I} + \frac{\sin|\boldsymbol{\omega}_{nb}^b \cdot \Delta t|}{|\boldsymbol{\omega}_{nb}^b \cdot \Delta t|} \cdot [\boldsymbol{\omega}_{nb}^b \cdot \Delta t \times] + \frac{1 - \cos|\boldsymbol{\omega}_{nb}^b \cdot \Delta t|}{|\boldsymbol{\omega}_{nb}^b \cdot \Delta t|^2} [\boldsymbol{\omega}_{nb}^b \cdot \Delta t \times]^2. \tag{3.35}$$

Eingesetzt in Gl. 3.33, mit $\boldsymbol{\sigma} = \boldsymbol{\omega}_{nb}^b \cdot \Delta t$:

$$\Delta v_s^n = \mathbf{R}_b^n(t_{k-1}) \left[\int_{t_{k-1}}^{t_k} \mathbf{f}^b \ \mathrm{dt} + \underbrace{\int_{t_{k-1}}^{t_k} \frac{\sin|\boldsymbol{\sigma}|}{|\boldsymbol{\sigma}|} \cdot [\boldsymbol{\sigma} \times] \ \mathrm{dt} \cdot \mathbf{f}^b}_{\mathbf{A}} + \underbrace{\int_{t_{k-1}}^{t_k} \frac{1 - \cos|\boldsymbol{\sigma}|}{|\boldsymbol{\sigma}|^2} [\boldsymbol{\sigma} \times]^2 \ \mathrm{dt} \cdot \mathbf{f}^b}_{\mathbf{B}} \right]. \tag{3.36}$$

Das Integral von **A** lautet streng:

$$\mathbf{A} = 2 \cdot \left[\boldsymbol{\omega}_{nb}^b \times \right] \cdot \left(\frac{\sin(\Delta t \cdot |\boldsymbol{\omega}_{nb}^b| \cdot \frac{1}{2})}{|\boldsymbol{\omega}_{nb}^b|} \right)^2. \tag{3.37}$$

Für kleine Werte von $\Delta t \cdot \boldsymbol{\omega}_{nb}^b$

$$\lim_{|\boldsymbol{\omega}_{nb}^b| \to 0} \frac{\sin(\Delta t \cdot |\boldsymbol{\omega}_{nb}^b| \cdot \frac{1}{2})}{|\boldsymbol{\omega}_{nb}^b|} = \frac{\Delta t}{2} \tag{3.38}$$

kann folgende Näherungslösung genutzt werden:

$$\mathbf{A} \approx \left[\boldsymbol{\omega}_{nb}^b \times \right] \cdot \frac{1}{2} \Delta t^2. \tag{3.39}$$

Im Gegensatz zu **A** konnte für das Integral von **B** im Rahmen dieser Arbeit keine geschlossene Lösung gefunden werden. Eine Reihenentwicklung dient als Ausweg:

$$\lim_{|\boldsymbol{\omega}_{nb}^b| \to 0} \frac{1 - \cos|\boldsymbol{\sigma}|}{|\boldsymbol{\sigma}|^2} = \frac{1}{2} \tag{3.40}$$

Tabelle 3.3.: Vergleich der Strapdown Algorithmen.

Korrektur	**Positionsfehler nach 180 Sek.**
Ohne Rotationskorrektur	15,96 m
Genäherte Rotationskorrektur **A** (Gl. 3.39)	2,41 m
Strenge Rotationskorrektur **A** (Gl. 3.37)	2,41 m
Strenge Rotationskorrektur **A** (Gl. 3.37) + **B** (Gl. 3.43)	2,38 m

$$\frac{1-\cos|\boldsymbol{\sigma}|}{|\boldsymbol{\sigma}|^2} = \frac{1}{2} - \frac{|\boldsymbol{\sigma}|^2}{24} + \frac{|\boldsymbol{\sigma}|^4}{720} - \frac{|\boldsymbol{\sigma}|^6}{40320} + \frac{|\boldsymbol{\sigma}|^8}{3628800} \cdots \tag{3.41}$$

$$\approx \frac{1}{2} - \frac{|\boldsymbol{\sigma}|^2}{24}. \tag{3.42}$$

Damit lässt sich das Integral für **B** berechnen:

$$\mathbf{B} \approx \left(\frac{\Delta t^3}{6} - \frac{|\boldsymbol{\omega}_{nb}^b|^2}{120} \cdot \Delta t^5 \right) \cdot \left[\boldsymbol{\omega}_{nb}^b \times \right]^2. \tag{3.43}$$

Die Geschwindigkeitsänderung für einen Zeitschritt lautet schließlich:

$$\Delta \mathbf{v}_s^n = \mathbf{R}_b^n(t) \cdot [\mathbf{I}\Delta t + \mathbf{A} + \mathbf{B}] \cdot \mathbf{f}^b. \tag{3.44}$$

Zur Verifikation wurde eine synthetische Kreis-Trajektorie aus der SIMA Software mit hoher Winkelgeschwindigkeit (1 rad/s) bei 100 Hz aufintegriert. Nach 180 Sekunden führt die Strapdown Integration zu den Positionsfehlern in Tabelle 3.3. Daraus lässt sich schließen, dass auf die Rotationskorrektur nicht verzichtet werden sollte. Wobei aber eine Korrektur über die Näherungslösung (Gl. 3.39) unter Verzicht auf **B** für viele Anwendungsfälle ausreichend ist.

3.4 Sensorfehler

Nach Grewal et al. (2007) haben folgende Fehler den größten Einfluss auf die Inertiale Navigationslösung:

- Offsetfehler (Bias)
- Skalierungsfehler (Scale)
- Unbekannte Variationen in den Offset- und Skalierungsfehlern (hauptsächlich von der Temperatur abhängig, aber auch Schwerkraft- bzw. Beschleunigungsabhängig)
- Misalignment (nicht-orthogonale Achsen)
- Weißes Rauschen
- Algorithmische Fehler (vgl. Abschnitt 3.3.1)

- Vibrationen (vgl. Abschnitt 4.6)

Sensorfehler lassen sich fast beliebig tiefgehend bzw. auf die Hardware hingehend modellieren und schätzen. Grewal und Andrews (2001) zeigen z. B. ein IMU Kalibriermodell mit 60 Unbekannten für hohe Genauigkeitsansprüche.

Eine einzelne IMU besteht intern aus zwei Einheiten: einem Gyroskop und einem Accelerometer. Pro Einheit sind drei näherungsweise orthogonal zueinanderstehende sensitiven Achsen verbaut. Eine der beiden Triaden (gekennzeichnet mit dem Subskript a) muss als datumsgebend ausgewählt werden. Für diesen Sensor a (beispielsweise das Accelerometer) gilt bei konstanter Temperatur:

$$\mathbf{l}_a^{\text{unkalib.}} + \mathbf{v} = \mathbf{M}_a \cdot \mathbf{S}_a \cdot \left(\tilde{\mathbf{l}}_a + \mathbf{b}_a\right) \tag{3.45}$$

mit der Skalenfehler-Matrix

$$\mathbf{S}_a = \begin{pmatrix} \alpha_x & 0 & 0 \\ 0 & \alpha_y & 0 \\ 0 & 0 & \alpha_z \end{pmatrix}, \tag{3.46}$$

der Sensor Misalignment-Matrix in Dreiecksform

$$\mathbf{M}_a = \begin{pmatrix} 1 & -\alpha_{yz} & \alpha_{zy} \\ 0 & 1 & -\alpha_{zx} \\ 0 & 0 & 1 \end{pmatrix} \tag{3.47}$$

und dem Bias Vektor

$$\mathbf{b}_a = \begin{pmatrix} (b_a)_x \\ (b_a)_y \\ (b_a)_z \end{pmatrix}. \tag{3.48}$$

Für den zweiten nicht-datumsgebenden Sensor b (beispielsweise das Gyroskop) gilt ebenfalls die Beobachtungsgleichung:

$$\mathbf{l}_b^{\text{unkalib.}} + \mathbf{v} = \mathbf{M}_b \cdot \mathbf{S}_b \left(\tilde{\mathbf{l}}_b + \mathbf{b}_b\right). \tag{3.49}$$

Mit dem Skalenfaktor:

$$\mathbf{S}_b = \begin{pmatrix} \beta_x & 0 & 0 \\ 0 & \beta_y & 0 \\ 0 & 0 & \beta_z \end{pmatrix}. \tag{3.50}$$

Wichtig ist nun, dass $\mathbf{M}_b$ eine vollbesetzte Matrix ist, da nun nicht nur die intrinsische Nicht-Orthogonalität der Achsen von Sensor b korrigiert werden muss, sondern auch die Orientierung gegenüber a:

$$\mathbf{M}_b = \begin{pmatrix} 1 & -\beta_{yz} & \beta_{zy} \\ \beta_{xy} & 1 & -\beta_{zx} \\ -\beta_{xy} & \beta_{yx} & 1 \end{pmatrix}. \tag{3.51}$$

Der Bias Vektor ist wieder einfach:

$$\mathbf{b}_b = \begin{pmatrix} (b_b)_x \\ (b_b)_y \\ (b_b)_z \end{pmatrix}. \tag{3.52}$$

Die Winkelfehler bzw. Abweichungen zur Orthogonalität werden als klein angenommen, was für alle marktüblichen Sensortriaden als gültig angesehen werden kann.

Es liegt also ein Ausgleichungsproblem mit klar definierten Beobachtungsgleichungen vor. Die Schwierigkeit bei der Ausgleichung liegt nun darin, dass für das Gyroskop der Vektor der wahren Messgrößen $\tilde{\mathbf{l}}$ in Bewegung nicht genau bekannt ist. Diese Bewegung wird aber benötigt, um die Skalenfehler und Misalignment Komponenten zu schätzen. Mit einem präzisen Roboterarm können die Drehraten vorgegeben werden. Die wahre Messgröße für die Accelerometerbeobachtungen können auf Stillstandsphasen reduziert werden (siehe Stillstandserkennung in Abschnitt 3.8), sodass die wahre Schwerebeschleunigung aus einem Modell bekannt ist.

Eine Möglichkeit teure Roboterarme für die Kalibrierung zu umgehen wird von Tedaldi et al. (2014) und Pretto und Grisetti (2014) vorgestellt. Die IMU muss manuell in verschiedene Lagen gebracht werden, dabei wird zwischen zwei Phasen unterschieden: Stillstandsphasen und Umorientierungsphasen. In den Stillstandsphasen wird das Accelerometer sowie der Bias Vektor des Gyroskops kalibriert. Mit Hilfe des kalibrierten Accelerometers können dann die Gyroskop Beobachtungen in den Umorientierungsphasen genutzt werden zur Kalibrierung des Gyroskops. Durch Aufintegrieren der Drehraten (siehe Gl. 3.19) müsste ein kalibriertes Gyroskop bei dem Erreichen der Stillstandsphase wieder die gleiche Orientierung erreichen wie das kalibrierte Accelerometer, welches im Stillstand die Schwerebeschleunigung misst. Der resultierende Winkelfehler kann dann wieder für ein Least-Squares Adjustment genutzt werden. Tedaldi et al. (2014) empfehlen 50 verschiedene Orientierungen und jeweils ca. 1-4 Sek. statische Perioden für eine zuverlässige Schätzung der Kalibrierparameter.

Hervorzuheben ist, dass gerade Low-Cost IMUs besonders temperaturempfindlich sind (Aggarwal et al., 2008). Schon kleine Temperaturänderungen können die beschriebenen Fehler signifikant ändern. Für eine höhere Genauigkeit ist es zu empfehlen, dass die beschriebene Kalibrierprozedur für verschiedene Temperaturen ausgeführt wird. Bei besonders hohen Anforderungen an z. B. die Beschleunigungsmessergenauigkeit in der Größenordnung $10^{-5}\, m/s^2$ sei hier auf Becker (2016) verwiesen. Im Betrieb kann dann aufgrund der tatsächlich vorliegenden Temperatur der richtige Kalibrierparametersatz angebracht werden bzw. aus den Kalibrierparametersätzen interpoliert werden. Restfehler sind unvermeidlich

und können durch die fortlaufende Zustandsschätzung kompensiert werden. Da die Fehlerschätzung im Zustandsvektor ein stochastisches Modell voraussetzt, ist es aber günstiger, wenn eben die temperaturabhängigen Fehler vorab korrigiert sind, dann müssen im stochastischen Modell keine zu pessimistischen Annahmen gemacht werden. Eine alternative Möglichkeit besteht in einer aktiven Temperaturregelung der Sensoren in einem geschützten Gehäuse. Ein Regler hält die Sensoren mit einem Heizelement bei einer bestimmten Temperatur, beispielsweise bei 50° C.

3.5 GNSS

Die Satellitennavigation kann als eines der wichtigsten Navigationssysteme angesehen werden, wenn nicht sogar als das wichtigste System für die Allgemeinheit. Für die Grundlagen sei an dieser Stelle auf die GPS/GNSS Standardwerke verwiesen: Hofmann-Wellenhof et al. (2008), Kaplan und Hegarty (2005), Parkinson et al. (1996) und Zarchan et al. (1996).

Die grundlegenden GNSS Messgrößen sind:

- Pseudorange (PSR) Messungen (*Code Pseudoranges*)
- Dopplermessungen (*Doppler Shift*)
- Trägerphasenmessungen (*Phase Pseudoranges*)

Darüber hinaus kann auch noch das geschätzte Signal-Rausch-Verhältnis (SNR) pro Satellit als Sensorinformation verstanden werden, sowie auch der Elevationswinkel der einzelnen Satelliten, beispielsweise für die Gewichtung der Messungen.

Die Auswertemöglichkeiten sind vielfältig. Aus Doppler- und Trägerphasenmessungen können genaue Geschwindigkeitsmessungen abgeleitet werden (Serrano et al., 2004), (Remondi, 2004), (Zwiener und Diekert, 2012); durch Bildung von Differenzen der Trägerphasenmessungen und Lösung der ganzzahligen Mehrdeutigkeiten können hochgenaue relative Positionierungen durchgeführt werden (Teunissen, 1995); durch das sogenannte Precise Point Positioning (PPP) können durch geeignete Modelle (Atmosphärische Einflüsse, Satellitenbahnen u. Uhrenfehler) die Beobachtungen signifikant verbessert werden (Kouba und Héroux, 2001), (Carcanague, 2013) oder (Schönemann, 2013). Space-based Augmentation Systeme (SBAS) können ebenfalls genutzt werden um GNSS Messungen zu korrigieren (Hofmann-Wellenhof et al., 2008).

Für die IMU/GNSS Integration beschreibt Jekeli (2001) folgende Ansätze: a) die dezentralisierte Verarbeitung (Position u. Geschwindigkeit) sowie b) die zentralisierte Verarbeitung (GNSS Rohdaten). Die dezentralisierte Verarbeitung fügt die Ergebnisse verschiedener Systeme zusammen. Jedes System hat seine eigene Signalverarbeitung. Bei einer zentralisierten Architektur werden alle Messungen in einem Master-Filter verarbeitet. Die zentralisierte Architektur ermöglicht die optimale Verarbeitung der Messungen. Auch bei weniger als vier Satelliten kann diese Art der Verarbeitung noch stützend wirken, siehe Zwiener (2012). Allerdings bietet die dezentrale Verarbeitung eine theoretisch vergleichbare Genauigkeit, solange die Satellitenkonstellation eine Positionslösung erlaubt und die Systemfehler über vollbesetzte Kovarianzmatrizen beschrieben werden. Darüber hinaus gibt es noch die sogenannte

Ultra-Tight und Deeply Coupled Integration. Hier unterstützt die IMU bzw. der geschätzte Navigationszustandsvektor die Signalverarbeitung des GNSS Empfängers, was besonders während widrigen Empfangsbedingungen (beispielsweise bei aktiver Störung der GNSS Signale sowie bei schwachem GNSS Signal) oder bei hoher Objektdynamik von Vorteil ist. Beim Ultra-Tight Coupling wird die geschätzte Geschwindigkeit in der Tracking-Loop genutzt, beim Deeply-Coupling wird der komplette Navigationszustandsvektor in die GNSS Signalverarbeitung und Regelkreise eingesetzt (Kiesel, 2012). Diese Verfahren benötigen einen kompletten Zugang zu einer GNSS Empfängerarchitektur und finden daher im Rahmen dieser Arbeit keine weitere Beachtung.

Hardwarenahe Aspekte im Bereich Hochfrequenztechnik und Antennenaufbau spielen eine nicht zu vernachlässigende Rolle bei der präzisen GNSS Navigation. Beispielsweise können sogenannte *Ground Planes* oder *Choke Rings* die Signalqualität steigern und den negativen Einfluss von Mehrwegeffekten lindern, siehe z. B. Zhang und Schwieger (2018). Schlechte Antennenverbindungen können die GNSS Messungen signifikant verschlechtern, auch wenn moderne Empfänger hier oft erstaunlich tolerant sind. Ein Indikator für eine schlechte Antennenverbindung ist für ein L1 Empfängersystem die Genauigkeit der Geschwindigkeitsmessung. Die Ungenauigkeit sollte im Bereich weniger cm/s liegen. Bei mehreren Dezimetern pro Sek. kann von einem Antennenproblem ausgegangen werden. Gute GNSS Empfänger haben zudem SAW-Filter[14] und (meist proprietäre) Algorithmen um Multipath Einflüsse zu minimieren (Carcanague, 2013). Zudem arbeiten Low-Cost Empfänger in den meisten Fällen nur auf der L1 Frequenz. Höherwertige Empfänger können zusätzlich auf der L2 bzw. L2C und L5-Frequenz messen, siehe dazu auch Becker (2009). Eine wichtige Rolle spielt auch die Stabilität der verbauten Oszillatoren. Beispiele für stabile Oszillatoren sind OCXO (Oven Controlled Crystal Oscillator) oder VCTCXO (Voltage Controlled Temperature Compensated Crystal Oscillator). Einfachere Empfänger verzichten auf Kompensationsmechanismen, was zu Frequenzabweichungen führt. Beispielsweise berichtet Realini (2009) bei Low-Cost Empfängern wie z. B. dem u-blox 4T von häufig auftretenden Cycle-Slips.

3.5.1 Pseudorange

Die Pseudorange (PSR) wird aus der Zeitdifferenz zwischen Empfänger r und Satellit s berechnet, siehe Abbildung 3.2 (Hofmann-Wellenhof et al., 2008).:

$$\mathrm{PSR}^{s,\mathrm{sys}} = c \cdot (T_r - T^s) \tag{3.53}$$

Die Zeitmessung wird u. a. verfälscht durch Uhrenfehler sowie atmosphärische Einflüsse:

$$\mathrm{PSR}^{s,\mathrm{sys}} = \sqrt{(x^s - x_r)^2 + (y^s - y_r)^2 + (z^s - z_r)^2} + c \cdot t_r - c \cdot t^s + c \cdot \tau^{\mathrm{sys}} + \Delta T^s + \Delta I_f^s \tag{3.54}$$

mit

- $^{\mathrm{sys}}$ GNSS abhängig (GPS, GLONASS, Galileo, Beidou ...)

[14] Akustische-Oberflächenwellen-Filter, also Bandpassfilter, die für GNSS relevanten Frequenzbereich optimiert sind. Engl.: *Surface Acoustic Wave Filter*

- T^s Sendezeit in Sekunden (aus Satellitenuhr)
- T_r Empfangszeit in Sekunden (Empfängeruhr)
- c Lichtgeschwindigkeit 299 792 458 m/s
- t_r Empfänger Uhrenfehler (gesucht)
- t^s Satelliten Uhrenfehler (inkl. relativistischer Korrekturen)
- ΔI_f^s Ionosphäreneinfluss (abhängig von Trägerphasenfrequenz f, Uhrzeit und Satellitenposition)
- ΔT^s Troposphäreneinfluss
- $(x^s, y^s, z^s)^T = \mathbf{r}^s$ Satellitenposition zum Sendezeitpunkt T^s (korrigiert nach Gl. 3.59)
- $(x_r, y_r, z_r)^T = \mathbf{r}_r$ Empfängerantennenposition zum Zeitpunkt T_r (gesucht)
- τ^{sys} Offset zwischen Systemzeit und Referenzsystemzeit

Bei dieser Darstellung wird angenommen, dass relativistische Effekte korrigiert wurden. Siehe dazu z. B. (Hofmann-Wellenhof et al., 2008, Kap. 5.4). Wird Gleichung (3.54) linearisiert, ergibt sich folgender Zusammenhang für jede Pseudorange-Beobachtung (pro Satellit):

$$\text{PSR}^{s,\text{sys}} = -\mathbf{e}_r^s \cdot \begin{pmatrix} \delta x \\ \delta y \\ \delta z \end{pmatrix} + l(\mathbf{r}_{r,0}, \mathbf{r}^s) + c \cdot t_r - c \cdot t^s + c \cdot \tau^{\text{sys}} + \Delta T^s + \Delta I_f^s. \tag{3.55}$$

$l(\mathbf{r}_{r,0}, \mathbf{r}^s)$ ist die aus einer Näherungsposition ($\mathbf{r}_{r,0}$) bestimmte Entfernung zwischen Antenne und Satellit. Als Verbesserungsgleichung kann man schreiben:

$$(\text{PSR}^{s,\text{sys}} - l(\mathbf{r}_{r,0}, \mathbf{r}^s)) + v = -\mathbf{e}_r^s \cdot \begin{pmatrix} \delta x \\ \delta y \\ \delta z \end{pmatrix} + c \cdot t_r - c \cdot t^s + c \cdot \tau^{\text{sys}} + \Delta T^s + \Delta I_f^s. \tag{3.56}$$

$\mathbf{e}_r^s$ ist der Einheitsvektor zwischen der aktuell bekannten Empfängerposition und der berechneten Satellitenposition:

$$\mathbf{e}_r^s = \frac{\mathbf{r}^s - \mathbf{r}_r}{|\mathbf{r}^s - \mathbf{r}_r|}. \tag{3.57}$$

Da sich die Erde zwischen dem Zeitpunkt T^s und T_r dreht, muss eine Korrektur an die Satellitenposition $\mathbf{r}^s$ angebracht werden. Dies ist im erdfesten System (ECEF) direkt über eine Rotation entlang der Z-Achse möglich. Bekannt auch als *Sagnac* Korrektur (Kaplan und Hegarty, 2005, S. 307):

$$\Delta\omega = \omega \cdot \frac{|\mathbf{r}^s - \mathbf{r}_r|}{c} \tag{3.58}$$

$$\mathbf{r}^s_{\text{Korrigiert}} = \begin{pmatrix} \cos\Delta\omega & \sin\Delta\omega & 0 \\ -\sin\Delta\omega & \cos\Delta\omega & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \mathbf{r}^s. \tag{3.59}$$

3.5.2 Phasenmessung

Für GNSS Phasenmessungen zum Zeitpunkt t kann folgende Beobachtungsgleichung pro Satellit s und Trägerphasenfrequenz f aufgestellt werden (Håkansson et al., 2017), (Hofmann-Wellenhof et al., 2008):

$$\begin{aligned} \Phi_f^{s,\text{sys}} = &\sqrt{(x^s - x_r)^2 + (y^s - y_r)^2 + (z^s - z_r)^2} \\ &+ c \cdot \left(t_r - t^s + b_f^{\text{sys}} - b_f^s + \tau^{\text{sys}} \right) \\ &- \Delta T^s - \Delta I_f^s + \lambda_f \cdot N_f^s + \lambda_f \cdot \omega^s + \text{ATX}_r^s, \end{aligned} \tag{3.60}$$

mit

- b_f^{sys} GNSS und frequenzabhängiger Receiverbias
- b_f^s Frequenzabhängiger Bias von Satelliten s
- ΔI_f^s Ionosphäreneinfluss (abhängig von Frequenz, Zeit u. Satellitenposition)
- λ_f Wellenlänge von Trägerphasenfrequenz f
- N_f^s Phasenmehrdeutigkeit pro Satellit (*ambiguity*), $N_f^s \in \mathbb{Z}$
- ω^s Carrier Phase Wind-up Korrektur[15]
- ATX_r^s Korrektur von richtungsabhängigen Variationen bezüglich des Phasenzentrums der Antenne, siehe z. B. Zeimetz et al. (2011)

In Form einer Verbesserungsgleichung kann folgender Zusammenhang formuliert werden:

$$\begin{aligned} \left(\Phi_f^{s,\text{sys}} - D^s(t) \cdot \lambda_f \right) + v = &\sqrt{(x^s - x_r)^2 + (y^s - y_r)^2 + (z^s - z_r)^2} \\ &+ c \cdot \left(t_r - t^s + b_f^{\text{sys}} - b_f^s + \tau^{\text{sys}} \right) \\ &- \Delta T^s - \Delta I_f^s + \lambda_f \cdot N_f^s + \lambda_f \cdot \omega^s + \text{ATX}_r^s. \end{aligned} \tag{3.61}$$

[15] Durch die relative Ausrichtung zwischen der Empfänger- und Satellitenantenne entsteht eine Phasenverschiebung, siehe Wu et al. (1992).

Dabei ist $D^s(t)$ der mitgezählte Doppler-Cyclecount, die Phasenmehrdeutigkeit N_f^s ist in diesem Zusammenhang als Anfangsunbekannte anzusehen.

3.5.3 Geschwindigkeit aus Dopplermessungen

Die Antennengeschwindigkeit am Punkt $\dot{\mathbf{r}}_r$ lässt sich durch die gemessene Dopplerfrequenzmessung f_{Doppler} berechnen. Pro Satellit s gilt folgende Beobachtungsgleichung (Kaplan und Hegarty, 2005):

$$\left(\mathbf{e}_r^s\right)^T \cdot \dot{\mathbf{r}}^s + c \cdot \left(\frac{f^s_{\text{Doppler,f}} - f}{f} + 1 \right) = \left(\mathbf{e}_r^s\right)^T \cdot \dot{\mathbf{r}}_r - c \cdot \dot{t}_r. \tag{3.62}$$

Der Vektor $\dot{\mathbf{r}}^s$ enthält die Satellitengeschwindigkeit. Die Berechnung aus den Broadcast Ephemeriden wird von Remondi (2004) beschrieben. f ist die Trägerfrequenz, $\dot{\mathbf{r}}_r$ die gesuchte Antennengeschwindigkeit und $\dot{t}_r$ der Empfängeruhrendrift. Die Empfängerposition muss auf ca. 10 m genau bekannt sein (Serrano et al., 2004).

3.5.4 Geschwindigkeit aus Trägerphasenmessungen

Gemessene Trägerphasen Φ (in Meter) lassen sich ebenfalls zur Bestimmung der Geschwindigkeit $\dot{\mathbf{r}}_r$ nutzen (Serrano et al., 2004):

$$\left(\mathbf{e}_r^s\right)^T \cdot \dot{\mathbf{r}}^s - \dot{\Phi}_f^s = \left(\mathbf{e}_r^s\right)^T \cdot \dot{\mathbf{r}}_r - c \cdot \dot{t}_r. \tag{3.63}$$

Dieses Verfahren ist präzise (im Bereich cm/s), allerdings ist mit Cycle-Slips in den Trägerphasenmessungen zu rechnen. Die Änderungsrate $\dot{\Phi}_f^s$ der Trägerphasenmessung ist z. B. numerisch zu berechnen:

$$\dot{\Phi}_f^s \approx \frac{\Phi^s_{t+1,f} - \Phi^s_{t-1,f}}{2 \cdot \Delta t}. \tag{3.64}$$

Nicht zu vernachlässigen ist der Hebelarm $\mathbf{o}^b$ zwischen der tatsächlich gesuchten Position $\mathbf{x}^n$ (das Zentrum des zu navigierenden Körpers) und der Position der GNSS Antenne. Der Hebelarm $\mathbf{o}^b$ wirkt sich hilfreich auf die Orientierungsschätzung aus, da die GNSS Beobachtung dadurch mit der Orientierungsschätzung in einem Zusammenhang steht. Umgekehrt wirkt sich ein vernachlässigter Hebelarm je nach Betrag und Bewegungsdynamik negativ auf die Zustandsschätzung aus. Für die kompletten formelmäßigen Zusammenhänge sei auf Abschnitt 4.3.2 verwiesen.

3.5.5 Precise Point Positioning (PPP) - nicht differentielle GNSS-Positionierung u. Geschwindigkeitsbestimmung

Pseudorangemessungen sowie Geschwindigkeitsmessungen bilden die Grundlage für die GNSS Stützung in Kapitel 5. Diese können durch PPP Techniken verbessert werden, siehe z. B. Kouba und Héroux (2001)

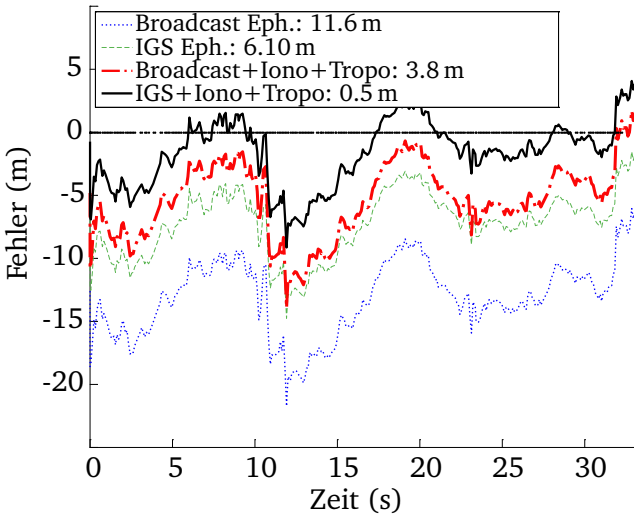
oder Carcanague (2013). Abbildung 3.9a zeigt den wahren Fehler einer GPS L1 Pseudorange[16]; die Pseudorange mit Broadcast Message Korrekturen hat hier im Mittel einen Fehler von 11,6 m. Wird der Satellitenuhrenfehler (t^s) und die Satellitenposition ($\mathbf{r}^s$) aus IGS Produkten berechnet, so reduziert sich der Pseudorangefehler auf 6,10 m. Wird zusätzlich durch die IGS Ionosphärenkarte (Abb. 3.9b) der Fehler ΔI korrigiert, reduziert sich der mittlere Pseudorangefehler insgesamt auf 0,5 m. Der IGS bietet sowohl Online-Korrekturdaten, als auch Offline Postprocessingdaten an (Dow et al., 2009). Für die Interpolation der Bahndaten ist das Lagrange Verfahren zu empfehlen (Hofmann-Wellenhof et al., 2008). Die TEC Maps sind als Single-Layer Modell berechnet. Der Kern des Single-Layer Modells basiert auf der Formel (Hofmann-Wellenhof et al., 2008):

$$\Delta I = \frac{1}{\cos\varphi} \cdot \frac{40.3}{f^2} \cdot \mathrm{TVEC} \tag{3.65}$$

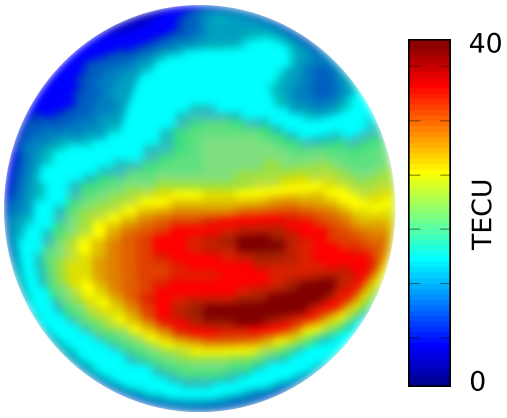
D. h. die Wegverzögerung ΔI in Metern lässt sich als Funktion der GNSS Frequenz in Hz, des Winkels φ und des TVEC Werts (Total Vertical Electron Count, Einheit TECU [17], entspricht etwa 16 cm Wegverzögerung) berechnen. Der Winkel φ lässt sich folgendermaßen berechnen:

$$\sin(\varphi) = \frac{R_E}{R_E + h_m} \sin(z_0) \tag{3.66}$$

Wobei z_0 die Zenitdistanz von Empfänger zu Satellit ist (90 Grad - Elevationswinkel), R_E der mittlere Erdradius und h_m die Höhe der Ionosphäre (typischerweise 300-400 km). Da die TEC Maps den Zustand zu einer bestimmten Uhrzeit darstellen, muss nicht nur innerhalb der Karte interpoliert werden, sondern auch zwischen zwei konsekutiven Epochen.



(a) Korrektur einer GPS L1 Pseudorange.



(b) Ionosphären TEC Karte.

Abbildung 3.9.: Precise Point Positioning Korrekturen.

[16] Empfängertyp: u-blox Antaris 4, rohdatenfähig.
[17] 1 TECU = 10^{16} Elektronen pro m^2

Durch eine zunehmende Verbreitung von GNSS Multifrequenzempfängern im Low-Cost Bereich, ist es zu erwarten, dass PPP Verfahren mehr und mehr zum Standard werden.

3.6 Barometrische Höhenmessung

Die barometrische Höhenmessung ist aufgrund der vergleichsweise günstigen und kompakten MEMS Barometer gut geeignet zur Stützung des instabilen Höhenkanals in einem Integrierten Navigationssystem (Gray und Maybeck, 1995). Da die horizontale Position durch die Schuler-Oszillation gestützt wird, reicht für ein Inertiales Navigationssystem bei hochwertigen Inertialsensoren die Integration eines Baro-Altimeters schon aus, um z. B. in der Luftfahrt eine Navigationslösung bereitzustellen. Im Low-Cost Bereich der Multikopternavigation sind Barometer längst etabliert. Ein Standalone Modell für die Multikopternavigation wird ausführlich von Meister (2010) beschrieben. Puls (2011) beschreibt einen weiteren möglichen Ansatz, in dem ein zweites Barometer als Bodenstation dient; dadurch können Änderungen im lokalen Luftdruck kompensiert werden, allerdings mit dem Nachteil, dass eine Bodenstation und eine permanente Funkverbindung notwendig ist. Die Änderungen im lokalen Luftdruck schwanken mit der Tageszeit und Wetterlage und resultieren in Höhenänderungen von z. T. mehreren Metern. Eine mögliche Mitigation dafür ist eine fortlaufende Biasschätzung mit Hilfe von GNSS Höhenmessungen (Meister, 2010), somit kann der Einfluss der Luftdruckschwankung korrigiert werden. Oder es kann auch einfach hingenommen werden, da z. B. bei Multikoptern die Flugzeit i. A. nicht ausreicht sowie bei manueller Steuerung unbewusst vom Piloten kompensiert wird. Die barometrische Höhe abgeleitet aus dem Luftdruck p (in Pascal) lässt sich nach Tanigawa et al. (2008) berechnen:

$$h(p) = 44300 \cdot \left(1 - \left(\frac{p}{p_0}\right)^{0,19}\right). \tag{3.67}$$

Bzw. in allgemeiner Form:

$$h(p) = \frac{T_0}{\left(\frac{-dT}{dH}\right)} \cdot \left(1 - \left(\frac{p}{p_0}\right)^{\left(\frac{-dT}{dH} \cdot \frac{R}{g}\right)}\right). \tag{3.68}$$

mit Variablen: Temperaturgradient $\frac{dT}{dH} = -6,5\,°\text{C/km}$, $p_0 = 101325\,\text{Pa}$, $T_0 = 288,15\,°\text{K}$, $g = 9,82\,\text{m/s}^2$ und $R = 287,052\,\text{m}^2/\text{s}^2/\text{K}$.

Besonders vorteilhaft ist die differentielle[18] Messung der barometrischen Höhen. Wenn keine absolute barometrische Höhe benötigt wird, kann der Höhenunterschied zwischen Startpunkt (Epoche 0) und aktueller Höhe (Epoche k) durch eine einfache Differenzenbildung bestimmt werden:

$$\Delta h_k = h(p_k) + b_e - h(p_0) - b_e \tag{3.69}$$

$$\Delta h_k = h(p_k) - h(p_0). \tag{3.70}$$

Der Vorteil ist, dass kein zweites Barometer am Boden benötigt wird, aber dennoch der Sensorfehler b_e eliminiert wird. Bei Barometern kann sich b_e durch Alterung verändern. Diese Fehler können aber über kurze Zeiträume als konstant angenommen werden und heben sich somit bei Gl. 3.69 auf.

[18] Nicht zu verwechseln mit der in der Luftfahrt üblichen Staudruckmessung mittels Pitotrohren.

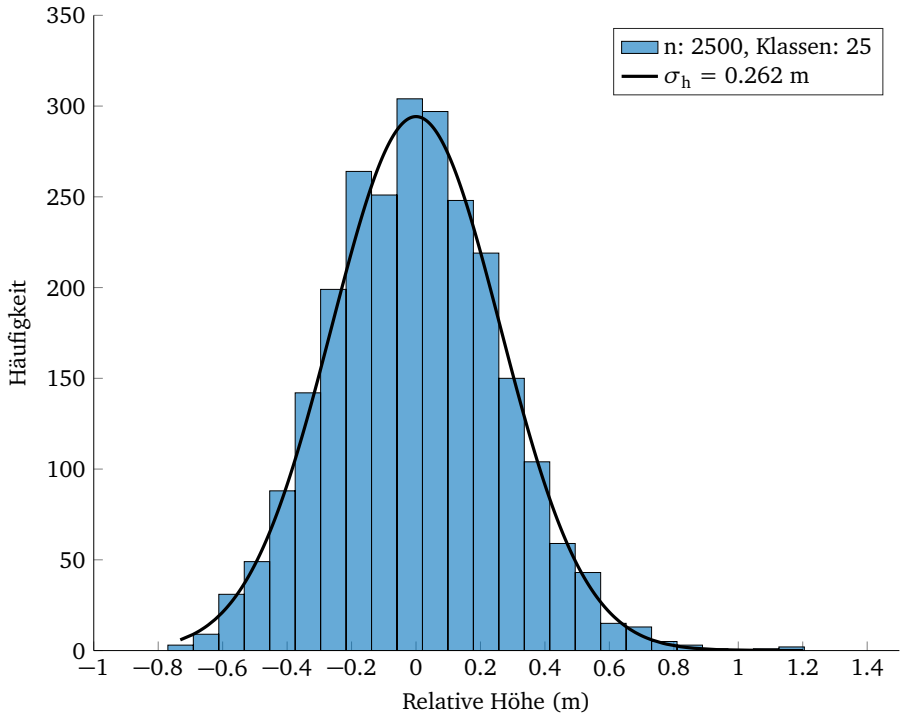


Abbildung 3.10.: Histogramm: Relative Höhe über Startpunkt aus Baro-Altimeter Messungen (aus MS5611 MEMS Barometer Beobachtungen).

Abbildung 3.10 zeigt die Verteilung von barometrischen Höhenmessungen bezogen auf den Startpunkt von einem *ruhenden* Barometer[19]. Werden die Daten aus Abbildung 3.10 dem Test auf Normalverteilung von Lilliefors (1967) unterzogen, so wird die Null Hypothese H_0:

$$h^b = \tilde{h}^b + v$$
$$H_0 : v \sim \mathcal{N}(0, \sigma_h),$$

bei einem Signifikanzniveau von $\alpha = 5\,\%$ angenommen. D. h. die Verteilung der dargestellten Barometermessungen unterscheidet sich nicht signifikant von einer Normalverteilung. Bei diesem weit verbreiteten Sensor kann somit von normalverteilten Messungen mit einer Streuung von $1\,\sigma = 0,26\,\text{m}$ ausgegangen werden, was die Zustandsschätzung erleichtert. Dennoch gibt es bei Multikoptern systematische Effekte, welche die Messungen verzerren können. Abschnitt 6.8 geht darauf gesondert ein. Darüber hinaus können in Gebäuden die Luftdruckmessungen zusätzlich durch das Öffnen von Türen, durch Aufzüge, Klimaanlagen, Tunnelfahrten oder offene Fenster bei einem Fahrzeug beeinflusst werden (Parviainen et al., 2008). Teile davon sind auf Temperaturänderungen zurückzuführen. Beobachtungsfehler von Low-Cost Barometer sind z. T. stark mit Temperaturänderungen korreliert. Bereits Temperaturänderungen von weniger als 1°C können bei einem MS5611 MEMS Barometer zu einem systematischen Höhenmessfehler im Dezimeterbereich führen (je nach Modell und Werkskalibrierung).

[19] Firma *Measurement Specialities* MS5611 MEMS Barometer, Werkskalibrierung aus PROM angebracht.

Da diese Störeinflüsse meist nicht von dauerhafter Natur sind, helfen hier robuste Schätzverfahren. Der Einfluss von Ausreißern in den barometrischen Messungen wird entsprechend Abschnitt 2.5 limitiert und kann insbesondere mit Hilfe der Inertialnavigation überbrückt werden.

3.7 Messung der magnetischen Flussdichte

Der Einsatz von MEMS IMUs mit geringer Güte bringt einige Herausforderungen mit sich, selbst wenn GNSS zur Stützung hinzugezogen wird. Im Stillstand ist die Azimutbestimmung für das Initial Alignment durch Gyrocompassing nicht möglich und bei nur einer GNSS Antenne kann die Satellitennavigation keine stützende Beobachtungen liefern. Magnetometer werden daher vor allem zur Azimutstützung in Navigationssystemen genutzt (Caruso, 1997). Blankenbach und Norrdine (2013) beschreiben darüber hinaus ein Positionierungssystem, das gezielt „*Magnetfeldstörungen*“ in Gebäuden mit Hilfe von Spulen aufbaut. Blankenbach und Norrdine (2013) geben erreichbare Positionsgenauigkeiten von unter 0,5 m in der Ebene an. Für die folgenden Kapitel soll aber der Fokus auf der Nordwinkelstützung liegen. Wendel et al. (2006) beschreiben ein Lösungskonzept, das auf Fluggeräte mit Low-Cost Sensorik zugeschnitten ist: ein Magnetometer kann direkt den Azimutwinkel für die Initialisierung bestimmen. Im Nominalbetrieb, also bei vorhandenem GNSS Signaltracking stützt das Magnetometer mit geringer Gewichtung weiterhin die Nordwinkelschätzung, was besonders in längeren Phasen ohne Beschleunigung auch notwendig ist. Dieses Konzept hat sich für das in Kapitel 4 vorgestellte Navigationssystem bewährt.

Ein Magnetometer liefert als Messung einen räumlichen Vektor in der Einheit Tesla, der SI-Einheit für die magnetische Flussdichte (gemessen im Sensor Koordinatensystem s, transformiert über das Plattform Koordinatensystem p in das körperfeste Koordinatensystem b):

$$\mathbf{l}^b = \mathbf{R}_p^b \cdot \mathbf{R}_s^p \cdot \begin{pmatrix} m_x^s \\ m_y^s \\ m_z^s \end{pmatrix}. \tag{3.71}$$

Die Annahme ist, dass diese Beobachtung durch das Magnetfeld der Erde dominiert wird, welches in Deutschland betragsmäßig bei ca. 48 µT liegt, mit einer jährlichen Änderungsrate der Deklination von ca. 0,1°/Jahr (Chulliat et al., 2015). Damit ist eine Schätzung des Azimuts und der Neigung (Pitch) gegenüber der Erde möglich – sofern die Position auf der Erde bekannt ist und ein Magnetfeldmodell vorliegt, vgl. Abb. 3.11. Wendel et al. (2006) empfehlen die Neigungsmessung durch ein Magnetometer zu ignorieren, da sie keine signifikanten Beiträge zur Zustandsschätzung liefert, sondern bei Störungen diese eher verschlechtert. Beispielsweise dürfen Geräte gemäß der *Verordnung über elektromagnetische Felder* (26. BImSchV) in Deutschland bei 50 Hz Störungen von 200 µT erzeugen. Somit ist (außer in kontrollierten Umgebungen) prinzipiell mit Ausreißern sowie langanhaltenden systematischen Störungen der Magnetometermessung zu rechnen[20]. Der Einsatz von robusten Schätzmethoden (vgl. Abschnitt 2.5) ist daher praktisch zwingend notwendig.

[20] Die Magnete im Teilchenbeschleuniger *LHC* am Schweizer Kernforschungszentrum CERN können gar ein Magnetfeld von 9 T (*ultimate field* genannt) erzeugen (Rossi, 2003).

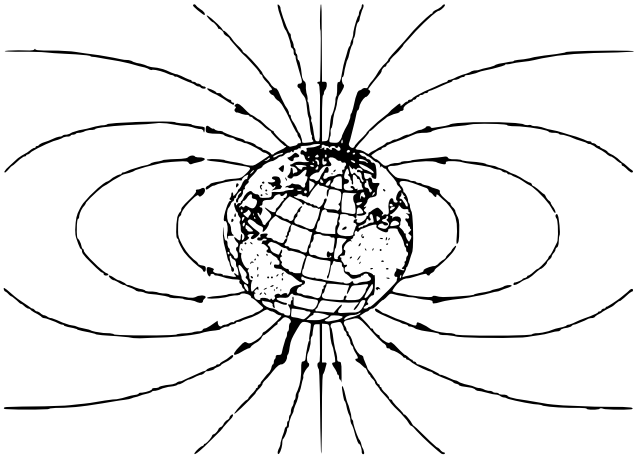


Abbildung 3.11.: Näherungsweise Darstellung des Magnetfelds der Erde (Dipolmodell), nach einer Abbildung von Caruso (1997).

Wie in Gl. 3.15 bereits beschrieben, lässt sich bei bekannter Ausrichtung des Magnetometers das Azimut durch Projektion der Messungen in die Ebene bestimmen. Die vollständigen mathematischen Zusammenhänge für die Zustandsschätzung sind in Abschnitt 4.3.6 zusammengefasst.

Essentiell für die Nutzbarkeit von Magnetometern ist eine Kalibrierung. Grundlegende Korrekturen der Bias und Skalenfehler sind in Caruso (1997) beschrieben, dieser Ansatz reicht aber i. A. nicht aus (Renaudin et al., 2010). Zur Korrektur von Hard- und Softiron Effekten sowie nicht-orthogonalen Sensorachsen, muss ein Ellipsoidfitting durchgeführt werden. Durch eine Orientierung des Sensors in möglichst allen Lagen werden Messpunkte auf allen drei Achsen gesammelt (X, Y, Z) und durch ein Best-Fit Ellipsoid modelliert, siehe Abbildung 3.12. In Abb. 3.12a sieht man als Punktwolke die gesammelten Messpunkte eines unkalibrierten Magnetometers[21], sowie ein Ellipsoid, das die Verteilung der Punkte beschreibt. In Abb. 3.12b sieht man die Daten nach der Kalibrierung, charakteristisch ist die Kugelform, die als Radius den Betrag der Erdmagnetfeldstärke (0,048 mT) hat und um den Ursprung des Koordinatensystems hin zentriert ist (biasfrei).

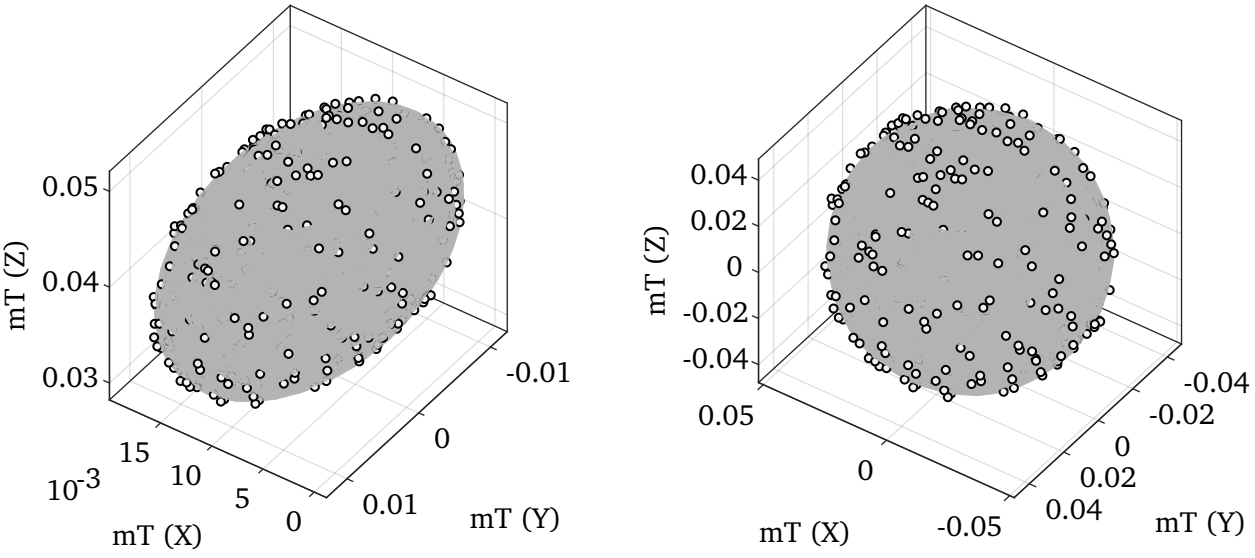
Das Fehlermodell ist im Prinzip analog zu den IMU Sensorfehlern (Abschnitt 3.4), bestehend aus einem 3×1 Biasfehlervektor $\mathbf{b}_m$ (enthält Sensorfehler sowie die Hard Iron Störungen), einer 3×3 Korrekturmatrix $\mathbf{M}_m$ (Sensorfehler sowie Soft Iron Störungen) sowie einer diagonal besetzten 3×3 Skalenfehlermatrix $\mathbf{S}_m$:

$$\mathbf{l}_m^{\text{unkalib.}} + \mathbf{v} = \mathbf{M}_m \cdot \mathbf{S}_m \left(\tilde{\mathbf{l}}_m + \mathbf{b}_m \right). \tag{3.72}$$

Merayo et al. (2000) beschreiben ein nicht-iteratives Ellipsoid Fitting Verfahren, welches auf Basis einer SVD-Zerlegung der VANDERMONDE Matrix die gesuchten Ellipsoid Parameter in einem Durchlauf bestimmt. Dieses Verfahren wurde u. a. genutzt für die Magnetometerkalibrierung der CHAMP Mission[22]. Die Kalibrierung in Abbildung 3.12 wurde auf Basis der Methode von Merayo et al. (2000) berechnet.

[21] Magnetometer: AKM AK8975.

[22] CHAMP - CHAllenging Minisatellite Payload, Satellitenmission unter Leitung des GFZ Potsdam von 2000–2010 zur Magnet- und Schwerefeldbestimmung der Erde.



(a) Unkalibrierte Messungen.

(b) Messungen nach Kalibrierung.

Abbildung 3.12.: Magnetometer Messpunkte (X,Y,Z) in Millitesla (mT). Ein Least-Squares Ellipsoid ist jeweils in die Punktwolke gelegt.

3.8 Zero Velocity Update

Ein sogenanntes Zero Velocity Update (ZUPT) ist eine einfache Möglichkeit um eine Navigationszustandsschätzung genauer zu machen und auch die Warm-Up Zeit des Filters zu verkürzen. Mit der Warm-Up Zeit ist die Zeit gemeint, bis die Zustandsvariablen mit einer akzeptablen Genauigkeit vorliegen. Ein ZUPT kann diese Zeit signifikant verkürzen. Populär wurde das Verfahren im Low-Cost Bereich u. a. durch eine Veröffentlichung von Foxlin (2005). Im Kern ist ein ZUPT nur eine Art fiktive Nullgeschwindigkeitsmessung, ohne dass tatsächlich ein solcher Sensor vorliegt[23]. Die praktische Umsetzung ist simpel, der Messvektor muss nur über eine Einheitsmatrix mit der geschätzten Geschwindigkeit in Verbindung gebracht werden. Ein einfaches Beispiel für einen Zustandsvektor, der aus einer 3D Position und Geschwindigkeit besteht ist in Gleichung 3.74 exemplarisch dargestellt.

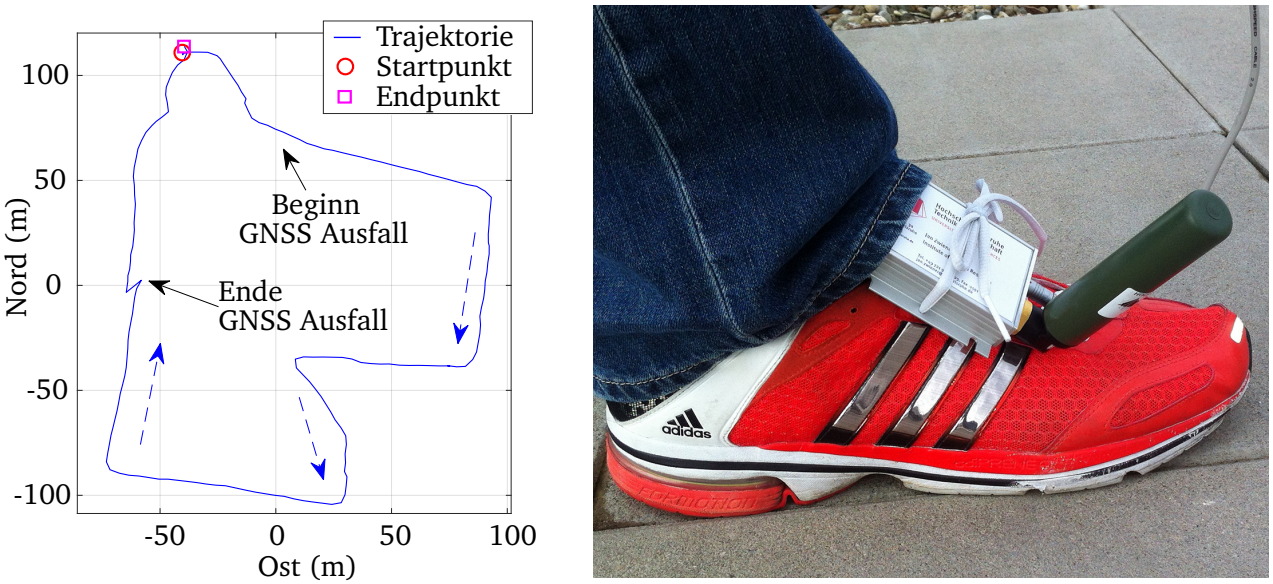
$$\mathbf{l}_{\text{ZUPT}} + \mathbf{v} = \mathbf{A} \cdot \mathbf{y} \tag{3.73}$$

$$\begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}_{\text{ZUPT}} + \mathbf{v} = \begin{pmatrix} \dots & \mathbf{I}_{3\times 3} & \dots \end{pmatrix} \cdot \begin{pmatrix} \vdots \\ \dot{\mathbf{x}} \\ \vdots \end{pmatrix}. \tag{3.74}$$

Dadurch, dass in einem Navigationsfilter praktisch immer die Vorhersagen auf Basis der aktuellen Geschwindigkeit und Beschleunigung durchgeführt werden, haben diese Variablen eine hohe Korrelation

[23] In Bebek et al. (2010) wird ein Kontaktsensor in einer Schuhsole für die Stillstandserkennung genutzt.

untereinander, was in der Kovarianzmatrix in den Nebendiagonalen abgebildet ist. Der vorteilhafte Effekt von der Einführung von Zero Velocity Updates ist somit nicht nur, dass einfach die Geschwindigkeit verbessert wird, sondern dass alle Zustandsvariablen entsprechend ihrer Korrelation verbessert werden. Das Verfahren wird in (Zwiener et al., 2014) im Detail für Foot-Mounted IMUs beschrieben, selbst bei



(a) Start und Ende des GNSS Signalverlusts.

(b) Foot-Mounted IMU u. GNSS Helix Antenne.

Abbildung 3.13.: Überbrückung eines GNSS Signalverlusts mit Zero Velocity Updates am Beispiel einer Foot-Mounted Navigationslösung.

MEMS Sensoren von geringer Güte, die normalerweise nach wenigen Sekunden Fehler im Meterbereich hätten, kann durch die ZUPT Stützung über lange Strecken hinweg ein Driften verhindert werden, siehe Abbildung 3.13. Für Anwendungen in der Luftfahrt sind Zero Velocity Updates im Flug nicht von Bedeutung, aber dafür vor dem Abheben sehr sinnvoll, um die IMU Biase schneller zu schätzen. Was gerade bei Fluggeräten mit Low-Cost IMUs besonders wichtig für die Flugstabilität ist.

Um einen Stillstand in einer Epoche k zu erkennen, müssen die vier Kriterien ($C1_k$, $C2_k$, $C3_k$ und $C4_k$) erfüllt sein, also die Summe muss 4 ergeben. Die Norm der Drehratenmessung muss unter einem Schwellwert t_ω liegen:

$$C1_k = \begin{cases} 1, & |\boldsymbol{\omega}_{ib}^b| < t_\omega \\ 0, & \text{sonst.} \end{cases} \tag{3.75}$$

Die Norm der Accelerometermessung $\mathbf{f}^b$ soll das Schwerefeld der Erde erfassen und darf keine tatsächlichen Beschleunigungen erfahren; somit soll f_{min} im Rahmen der Messgenauigkeit unter 9.81 m/s² und f_{max} über 9.81 m/s² liegen.

$$C2_k = \begin{cases} 1, & f_{min} < |\mathbf{f}^b| < f_{max} \\ 0, & \text{sonst.} \end{cases} \tag{3.76}$$

Die Standardabweichung $\mathbf{s_f}$ der Accelerometermessungen $\mathbf{f}^b$ muss unter einem bestimmtem Schwellwert σ_a liegen. Das Intervall für die Berechnung der Standardabweichung hängt von der Objektdynamik ab. Für Multikopter- oder Fahrzeuganwendungen sind 100 ms ein empfehlenswerter Bereich.

$$C3_k = \begin{cases} 1, & \mathbf{s_f} < \sigma_a \\ 0, & \text{sonst.} \end{cases} \tag{3.77}$$

Die Standardabweichung $\mathbf{s_\omega}$ der Drehratenmessungen $\boldsymbol{\omega}_{ib}^b$ muss unter einem bestimmtem Schwellwert σ_ω liegen.

$$C4_k = \begin{cases} 1, & \mathbf{s_\omega} < \sigma_\omega \\ 0, & \text{sonst.} \end{cases} \tag{3.78}$$

3.9 Zero Rotation Update

Die sogenannten Zero Rotation Updates (ZRU) sind auf den ersten Blick ähnlich zu den Zero Velocity Updates (ZUPT), aber dennoch im Detail unterschiedlich und für höchste Genauigkeit getrennt zu betrachten. Zero Rotation Updates können direkt genutzt werden um den Gyroskop Bias Fehler $\mathbf{b}_\omega$ zu schätzen. In totaler Ruhelage kann dann die Drehratenmessung $\boldsymbol{\omega}_{nb}^n$ direkt als Bias-Beobachtung in die Ausgleichung fließen:

$$\mathbf{l}_{\text{ZRU}} = \mathbf{A} \cdot \mathbf{y} \tag{3.79}$$

$$\boldsymbol{\omega}_{nb}^n = \begin{pmatrix} \omega_x \\ \omega_y \\ \omega_z \end{pmatrix}_{\text{ZRU}} = \begin{pmatrix} \dots & \mathbf{I}_{3\times 3} & \dots \end{pmatrix} \cdot \begin{pmatrix} \vdots \\ \\ \mathbf{b}_\omega \\ \\ \vdots \end{pmatrix}. \tag{3.80}$$

Für die ZRU Erkennung werden nur Drehraten betrachtet und die Varianz s_ω^2 mit der Redundanz $r_1 = n - u$ (Anzahl Messungen n - Unbekannte u) über ein Zeitintervall t_{ZRU} berechnet. Die wahre Varianz (σ_ω^2) muss für den Sensor vorab ermittelt worden sein. Ob für das Zeitintervall dann ein Zero Rotation Update durchgeführt werden kann, ergibt sich aus einem Fisher Test, der dann für alle Achsen durchgeführt wird (Zwiener et al., 2014) u. (Jäger et al., 2005):

Null Hypothese:

$$E[s_\omega^2] = \sigma_\omega^2 \tag{3.81}$$

Alternativhypothese:

$$E[s_{\omega}^2] > \sigma_{\omega}^2 \tag{3.82}$$

Testgröße:

$$F = \frac{s_{\omega}^2}{\sigma_{\omega}^2} \tag{3.83}$$

Als Quantil mit dem Signfikanzniveau α:

$$F_{r_1,\infty,1-\alpha}. \tag{3.84}$$

Liegt auf einer der drei Achsen die Testgröße über dem kritischen Wert, so stimmt die Varianz der Messwerte nicht mehr mit der erwarteten Varianz in Ruhe überein und es kann von einer Bewegung ausgegangen werden.

4 Navigation für Luftfahrzeuge mit verteilten Sensoren

4.1 Navigationszustandsschätzung mit Hilfe eines Kalman-Filters

Die Anwendung des linearisierten Kalman-Filters (siehe Abschnitt 2.9.2) auf Navigationsprobleme in dieser Arbeit basiert auf den Beiträgen von Wendel (2011), Wendel et al. (2006), Meister (2010), Farrell (2008), Titterton und Weston (2004), Britting (1971) sowie z. T. Grewal und Andrews (2001). Im Prinzip bauen alle genannten Werke auf dem gleichen Error-State Kalman-Filter Ansatz auf und können als *State of the Art* betrachtet werden. Auf die komplette Herleitung der Ableitungen wird verzichtet, diese sind in den genannten Standardwerken zu finden.

Die Parametrisierung findet im n-Frame statt, die Orientierung wird mit Hilfe von geschätzten kleinen Winkelfehlern (Gl. 4.5) korrigiert und außerhalb des Error-State Vektors in Form eines Quaternion mitgeführt. Es soll im n-Frame navigiert werden, somit bildet Navigationsgleichung 3.21 die Grundlage. Es wird eine initiale Navigationszustandsschätzung für die Position, Geschwindigkeit und Orientierung benötigt. Darüber hinaus werden als Hilfsparameter die Gyroskop und Accelerometer Biasvektoren fortlaufend im Zustandsvektor geschätzt (Online-Kalibrierung). Die Erfahrung zeigt, dass eine Kalibrierung der übrigen Sensorfehler (siehe Abschnitt 3.4) in einer Offline-Kalibrierung bei aktuellen Low-Cost MEMS IMUs ausreichend ist. Der IMU Bias sollte zwar auch Offline vorkalibriert werden (gerade Low-Cost Sensoren haben hier oft erstaunlich große Offset-Fehler), aber eine Online-Kalibrierung der Biasfehler ist aufgrund von variablen Fehleranteilen notwendig. Zu diesen Restfehlern zählt: a) der *turn-on bias* sowie b) ein über die Zeit variabler Biasfehler *time variant bias* (Bancroft, 2010). Für die Beobachtungen in den folgenden Abschnitten wird angenommen, dass die Offline-Kalibrierungen vorab angebracht wurden.

Es soll folgender 17×1 Error-State geschätzt werden. Dazu sei nochmal auf den in Gl. 2.93 beschriebenen Ansatz verwiesen. Dieser Vektor kann als Abweichung der Vorhersage gegenüber einem Teil eines kompletten Navigationszustandsvektor (siehe Gl. 3.5) verstanden werden, wobei hier beispielsweise die Beschleunigung $\ddot{x}$, Drehraten ω oder die Rateänderungen $\dot{\omega}$ fehlen, dafür werden Zusatzparameter $\delta\mathbf{z}_1$ zur Biasschätzung eingeführt. Diese Abweichungen gehen gegen 0 und sind keine Parameter im eigentlichen Sinne. Die Drehraten u. Beschleunigungen werden also nicht im Zustandsvektor geschätzt, sondern direkt als kalibrierte Sensorbeobachtungen im Strapdown Algorithmus (siehe Abschnitt 3.3) verarbeitet.

Um rohe GNSS Beobachtungen eng in das Filter zu integrieren, wird der GNSS Empfängeruhrenfehler t_r (inkl. Drift $\dot{t_r}$) zusätzlich geschätzt:

$$\delta\bar{\mathbf{y}}_k^+ = \left(\begin{array}{c} \delta\mathbf{x} \\ \delta\dot{\mathbf{x}} \\ \delta\boldsymbol{\psi} \\ \hline \delta\mathbf{b}_a \\ \delta\mathbf{b}_\omega \\ \delta t_r \\ \delta\dot{t}_r \end{array}\right) \tag{4.1}$$

bzw. in Kurzform zusammengefasst zu einem reduzierten Zustandsvektor $\mathbf{y}_1$ (gegenüber Gl. 3.5) u. Zusatzparametern $\mathbf{z}_1$ gegenüber Gl. 3.5.

$$\delta\bar{\mathbf{y}}_k^+ = \left(\frac{\delta\mathbf{y}_1^+}{\delta\mathbf{z}_1^+}\right). \tag{4.2}$$

Bestehend aus den

- $\delta\mathbf{x}$ 3×1 Positionsfehlervektor
- $\delta\dot{\mathbf{x}}$ 3×1 Geschwindigkeitsfehlervektor
- $\delta\boldsymbol{\psi}$ 3×1 Orientierungsfehlervektor (zu beachten, dass hier kein 4×1 Quaternion vorliegt)
- $\delta\mathbf{b}_a$ 3×1 Accelerometerbiasvektor
- $\delta\mathbf{b}_\omega$ 3×1 Gyroskopbiasvektor
- δt_r Skalarer Fehler in GNSS Empfängeruhrenfehler, skaliert mit der Lichtgeschwindigkeit c
- $\delta\dot{t}_r$ Skalarer Fehler in GNSS Empfängeruhrendrift, ebenfalls skaliert mit c

Falls keine rohen GNSS Beobachtungen integriert werden, kann dieser Error-State Vektor einfach auf einen 15×1 Vektor reduziert werden:

$$\delta\bar{\mathbf{y}}_k^+ = \left(\begin{array}{c} \delta\mathbf{x} \\ \delta\dot{\mathbf{x}} \\ \delta\boldsymbol{\psi} \\ \hline \delta\mathbf{b}_a \\ \delta\mathbf{b}_\omega \end{array}\right). \tag{4.3}$$

bzw. wieder zusammengefasst im Vergleich zu Gl. 3.5:

$$\delta\bar{\mathbf{y}}_k^+ = \left[\frac{d\mathbf{y}_1^+}{d\mathbf{z}_1^+}\right]. \tag{4.4}$$

Es wird davon ausgegangen, dass die Sensor-Offsets mit der Zeit variieren (beispielsweise durch Temperaturänderungen oder Vibrationen), sodass das zeitliche Verhalten als Random-Walk-Prozess modelliert wird (vgl. Abschnitt 2.10). Von besonderer Bedeutung ist die Darstellung der Orientierungsfehler $\delta\boldsymbol{\psi}$. Diese sind nicht mit Euler-Winkeln zu verwechseln. Unter der Annahme, dass die Winkelfehler $\delta\phi, \delta\theta$ und $\delta\psi$ klein sind, kann eine Drehmatrix auf den Grundlagen der Kleinwinkelnäherung[1] ausgedrückt werden (siehe auch Gl. 2.146):

$$\mathbf{R}_n^{\hat{n}} = \begin{pmatrix} 1 & -\delta\psi & \delta\theta \\ \delta\psi & 1 & -\delta\phi \\ -\delta\theta & \delta\phi & 1 \end{pmatrix}. \tag{4.5}$$

Diese Darstellung der Orientierung macht die Error-State Darstellung für die Navigation attraktiv. Wie in den folgenden Abschnitten gezeigt wird, ergeben sich damit lineare bzw. quasi-lineare Zusammenhänge, sodass das Kalman-Filter seine Stärken ausspielen kann. Vorteilhaft ist zudem, dass keine Zwangsbedingungen eingeführt werden müssen, wie es etwa bei einem 4×1 Quaternion[2] Zustandsvektor der Fall wäre.

Eine erweiterte Anmerkung verdient noch der geschätzte GNSS Uhrenfehler δt_r: Bei der Integration von verschiedenen Satellitennavigationssystemen ist zu beachten, dass jedes System ein eigenes Zeitsystem hat. Die aktuellen Umrechnungsparameter werden durch die jeweilige Broadcast Message übermittelt und müssen algorithmisch angebracht werden (Hofmann-Wellenhof et al., 2008). Selbst kleinste Uhrenfehler wirken sich allgemein jedoch bei Satellitennavigationssystemen sehr sensibel auf die Positionslösung aus. Daher kann darüber hinaus noch ein Restuhrenfehler τ^{sys} zwischen der GPS Zeit und dem jeweiligen System (z. B. Galileo, GLONASS oder Beidou) als weiterer *Nuisance* Parameter mitgeschätzt werden (Borre et al., 2007). Hier kann aber keine allgemeingültige Empfehlung gegeben werden, da auch die Hersteller der GNSS Receiver die Inter-System Uhrenfehler unterschiedlich behandeln. Viele Hersteller korrigieren periodisch immer wieder den Uhrenfehler, andere Hersteller wiederum korrigieren kontiniuerlich. Uhrenfehlersprünge müssen algorithmisch detektiert und korrigiert werden. Es sei an der Stelle auf einen praxisnahen Beitrag von Petovello (2011) verwiesen.

[1] $\sin\alpha \approx \alpha$, $\cos\alpha \approx 1$ für kleine Werte von α.

[2] Die Länge des Quaternions muss genau 1 betragen, da sonst die resultierenden Transformationen nicht mehr Längen- und Winkeltreu sind (Lengyel, 2004).

4.2 Vorhersagemodell

Aufbauend auf Gl. 2.133 wird nun ein Gleichungssystem aufgestellt, das die zeitliche Änderung der Zustandsfehler beschreibt. Dieses Gleichungssystem wird um die stochastische Modellierung gemäß Gl. 2.124 erweitert. Nach einer Herleitung von Wendel (2011):

$$\frac{\partial}{\partial \mathrm{t}} \underbrace{\begin{pmatrix} \delta\mathbf{x} \\ \delta\dot{\mathbf{x}} \\ \delta\boldsymbol{\psi} \\ \hline \delta\mathbf{b}_a \\ \delta\mathbf{b}_\omega \\ \delta t \\ \delta\dot{t} \end{pmatrix}}_{\delta\bar{\mathbf{y}}} = \underbrace{\begin{bmatrix} \mathbf{F}_{11} & \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} & 0 & 0 \\ \mathbf{F}_{21} & \mathbf{F}_{22} & \mathbf{F}_{23} & -\hat{\mathbf{R}}_b^n & \mathbf{0} & 0 & 0 \\ \mathbf{F}_{31} & \mathbf{F}_{32} & \mathbf{F}_{33} & \mathbf{0} & -\hat{\mathbf{R}}_b^n & 0 & 0 \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & 0 & 0 \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & 0 & 0 \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & 0 & 1 \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & 0 & 0 \end{bmatrix}}_{\mathbf{F}} \cdot \underbrace{\begin{pmatrix} \delta\mathbf{x} \\ \delta\dot{\mathbf{x}} \\ \delta\boldsymbol{\psi} \\ \hline \delta\mathbf{b}_a \\ \delta\mathbf{b}_\omega \\ \delta t \\ \delta\dot{t} \end{pmatrix}}_{\delta\bar{\mathbf{y}}} + \underbrace{\begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & 0 & 0 \\ \hat{\mathbf{R}}_b^n & \mathbf{0} & \mathbf{0} & \mathbf{0} & 0 & 0 \\ \mathbf{0} & \hat{\mathbf{R}}_b^n & \mathbf{0} & \mathbf{0} & 0 & 0 \\ \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} & 0 & 0 \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} & 0 & 0 \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & 1 & 0 \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & 0 & 1 \end{bmatrix}}_{\mathbf{G}} \cdot \underbrace{\begin{pmatrix} \mathbf{n}_a \\ \mathbf{n}_\omega \\ \mathbf{n}_{b_a} \\ \mathbf{n}_{b_\omega} \\ n_t \\ n_{\dot{t}} \end{pmatrix}}_{\mathbf{w}} \tag{4.6}$$

mit:

- $\hat{\mathbf{R}}_b^n$ 3×3 Geschätzte Transformationsmatrix, b-Frame zu n-Frame
- $\mathbf{F}_{ii}$ 3×3 Submatrix
- $\mathbf{0}$ 3×3 Nullmatrix
- $\mathbf{I}$ 3×3 Einheitsmatrix
- $\mathbf{n}_a$ 3×1 Zeitkontinuierliches weißes Accelerometer Systemrauschen in der Einheit $\frac{m/s^2}{\sqrt{\mathrm{Hz}}}$
- $\mathbf{n}_\omega$ 3×1 Zeitkontinuierliches weißes Gyroskop Systemrauschen in der Einheit $\frac{rad/s}{\sqrt{\mathrm{Hz}}}$
- $\mathbf{n}_{b_a}$ 3×1 Accelerometer Bias Random Walk in der Einheit $\frac{m/s^3}{\sqrt{\mathrm{Hz}}}$
- $\mathbf{n}_{b_\omega}$ 3×1 Gyroskop Bias Random Walk in der Einheit $\frac{rad/s^2}{\sqrt{\mathrm{Hz}}}$
- n_t Weißes Rauschen f. GNSS Empfängeruhrenfehler
- $n_{\dot{t}}$ Weißes Rauschen f. GNSS Empfängeruhrenfehlerdrift

Die stochastische Modellierung der Sensorfehler wird in Abschnitt 2.11 behandelt; der Einfluss von Vibrationen hat darüber hinaus noch einen Einfluss auf diese Fehler. Dies wird in Abschnitt 4.6 exemplarisch betrachtet. Für Low-Cost MEMS IMU Anwendungen konnte kein Performanceverlust festgestellt werden, wenn folgende Matrizen zu 0 gesetzt werden[3]:

$$\mathbf{F}_{11} = \mathbf{F}_{21} = \mathbf{F}_{31} = \mathbf{F}_{22} = \mathbf{F}_{32} = \mathbf{F}_{33} = \mathbf{0}. \tag{4.7}$$

[3] Für die vollständigen Submatrizen $\mathbf{F}_{ii}$ bei hochwertiger Sensorik sei auf Kap. 8.2 in Wendel (2011) verwiesen.

Nach dieser Vereinfachung kann die Vorhersagedifferentialgleichung folgendermaßen interpretiert werden: ein Fehler in der Geschwindigkeit $\delta\dot{\mathbf{x}}$ führt zukünftig zu einem Fehler in der Position (Zeile 1 in Matrix $\mathbf{F}$), ein Fehler im Bias des Accelerometers $\delta\mathbf{b}_a$ führt zu einem Geschwindigkeitsfehler (Zeile 2), ein Fehler im Gyroskop Bias $\delta\mathbf{b}_\omega$ führt zu einem Lagefehler.

Besonders ist die Matrix $\mathbf{F}_{23}$ hervorzuheben:

$$\mathbf{F}_{23} = -\left[\left(\mathbf{R}_b^n \cdot \hat{\mathbf{f}}_{ib}^b\right)\times\right] = \begin{pmatrix} 0 & (f_{ib}^n)_z & -(f_{ib}^n)_y \\ -(f_{ib}^n)_z & 0 & (f_{ib}^n)_x \\ (f_{ib}^n)_y & -(f_{ib}^n)_x & 0 \end{pmatrix}. \tag{4.8}$$

Die aktuelle Schätzung der Specific Force $\hat{\mathbf{f}}_{ib}^b$ wird in den n-Frame transformiert und in eine kreuzproduktbildende Matrix überführt. Das Systemmodell über Matrix $\mathbf{F}_{23}$ besagt im Prinzip, dass ein Fehler in der Orientierung ($\delta\boldsymbol{\psi}$) zu einem Fehler in der angenommenen Beschleunigung $\ddot{\mathbf{x}}$ führt. Diese Beziehung erzeugt für die Zustandsschätzung über die Zeit hinweg eine Korrelation zwischen der Position, Geschwindigkeit und der Orientierung. Dazu muss allerdings auch eine Beschleunigung vorliegen. Wenn z. B. in der Ebene keine Beschleunigungen vorliegen, also während Stillstandsphasen und bei konstanter Geschwindigkeit, kann das Heading (Yaw) nicht gestützt werden. Wenn das vorgestellte Filter ohne Magnetometerstützung in einer Autofahrt eingesetzt wird, kann dies einfach getestet werden: nach längerer Fahrt mit einer konstanten Geschwindigkeit baut sich ein Nordwinkelfehler auf, der innerhalb weniger Augenblicke durch ein leichtes Abbremsen oder Beschleunigen des Fahrzeugs korrigiert werden kann. Das gleiche gilt natürlich für Fluggeräte, jede Art von Trajektoriendynamik unterstützt die Zustandsmessung. Selbst bei einer um 90° falschen Anfangsorientierung kann durch ausreichende Beschleunigung der Nordwinkel auch ohne ein Magnetometer geschätzt werden. Was auch zeigt, dass die Approximation über kleine Winkelfehler sehr robust ist[4].

Der Vektor $\delta\mathbf{y}^-$ muss, wie in Abschnitt 2.9.2 beschrieben, nicht explizit berechnet werden (Becker, 2016). Nach Gleichung 2.93

$$\mathbf{y}_k^+ = \mathbf{y}_{k,0}^+ + \delta\mathbf{y}_k^+ \tag{4.9}$$

bzw.

$$\mathbf{y}_{k,0}^+ = \mathbf{y}_k^- \tag{4.10}$$

ergibt sich somit:

$$\delta\mathbf{y}^- = \mathbf{0}. \tag{4.11}$$

[4] Für die Fahrzeugnavigation bietet sich ein Soft-Constraint des Yaw-Winkels über Geschwindigkeitsmessungen an, bei dem angenommen wird, dass sich das Fahrzeug in Richtung der Roll-Achse bewegt. Eine derartige Bedingung kann ein Magnetometer überflüssig machen. Dies wird auch *Automotive-Mode* genannt.

Tabelle 4.1.: Euler-Vorwärts Gleichungen zur Vorhersage des Navigationszustandsvektors

Komponente	Gleichung	Details
$\mathbf{x}^n$	$\mathbf{x}^n(t+\Delta t) = \mathbf{x}^n(t) + \dot{\mathbf{x}}^n(t) \cdot \Delta t$	siehe Gl. 3.27
$\dot{\mathbf{x}}^n$	$\dot{\mathbf{x}}^n(t+\Delta t) = \dot{\mathbf{x}}^n(t) + \Delta \mathbf{v}^n$ mit $\Delta \mathbf{v}^n = \ddot{\mathbf{x}}^n \cdot \Delta t +$ Rotationskorr.	siehe Gl. 3.24, 3.25
$\ddot{\mathbf{x}}^n$		Details, siehe Gl. 3.21, 3.25
ψ bzw. $\mathbf{q}_b^n$	$\mathbf{q}_b^n(t+\Delta t) = \mathbf{q}_b^n(t) \bullet \begin{pmatrix} \cos(\frac{1}{2} \cdot \lvert\boldsymbol{\omega}_{nb}^b\rvert \cdot \Delta t) \\ \frac{1}{\lvert\boldsymbol{\omega}_{nb}^b \cdot \Delta t\rvert} \cdot \boldsymbol{\omega}_{nb}^b \cdot \Delta t \cdot \sin(\frac{1}{2} \cdot \lvert\boldsymbol{\omega}_{nb}^b\rvert \cdot \Delta t) \end{pmatrix}$	siehe Gl. 3.19
$\mathbf{b}_a$	$\mathbf{b}_a(t+\Delta t) = \mathbf{b}_a(t)$	
$\mathbf{b}_\omega$	$\mathbf{b}_\omega(t+\Delta t) = \mathbf{b}_\omega(t)$	
t_{GNSS}	$t_{\text{GNSS}}(t+\Delta t) = t_{\text{GNSS}}(t) + \dot{t}_{\text{GNSS}} \cdot \Delta t$	
$\dot{t}_{\text{GNSS}}$	$\dot{t}_{\text{GNSS}}(t+\Delta t) = \dot{t}_{\text{GNSS}}(t)$	

Die Vorhersage wird durch numerische Integration der entsprechenden Differentialgleichungen durchgeführt, siehe Abschnitt 3.3. D. h. für die Elemente des Navigationszustandsvektors:

- Position u. Geschwindigkeit: Lösen der Navigationsgleichungen (Hofmann-Wellenhof et al., 2003, Kap. 11.3.3). Die Gleichungen werden in Abschnitt 3.3.1 betrachtet.
- Beschleunigung: wird nicht explizit im Navigationszustandsvektor geschätzt. Eine epochenweise Behandlung erfolgt als sogenannter *System-Input* mit Gl. 3.25a u. 3.25b, unter Berücksichtigung der aktuellen Biasschätzung $\mathbf{b}_a$.
- Orientierung: Direkte Lösung über Gl. 3.19.
- Hilfsvariablen (Bias, GNSS Uhrenfehler): Modellierung als Random-Walk.

Um die Übersichtlichkeit zu erhöhen, sind die genannten Formeln in Tabelle 4.1 in Kurzform für das Euler-Vorwärts-Verfahren wiederholt dargestellt.

Die Unsicherheiten, die durch diesen Schritt entstehen, müssen aber modelliert werden. Dieser stochastische Teil der Vorhersage im zeitdiskreten Bereich über das Zeitintervall Δt von Epoche $k-1$ zu k lässt sich wie folgt berechnen[5]:

$$\boldsymbol{\Phi} = \mathbf{I} + \mathbf{F} \cdot \Delta t + \frac{1}{2} \cdot (\mathbf{F} \cdot \Delta t)^2 + h.o.t. \tag{4.12}$$

$$\mathbf{Q}_{yy,k}^- = \boldsymbol{\Phi} \cdot \mathbf{Q}_{yy,k-1} \cdot \boldsymbol{\Phi}^T + \mathbf{G} \cdot \mathbf{Q} \cdot \mathbf{G}^T \cdot \Delta t. \tag{4.13}$$

Die Matrix $\mathbf{Q}$ beschreibt stochastisch den Vektor $\mathbf{w} \sim \mathcal{N}(0, \mathbf{Q})$ aus Gl. 4.6. Mit *h.o.t.* sind Terme höherer Ordnung gemeint, die an dieser Stelle vernachlässigt werden.

Die Zeit Δt ist nicht von der IMU Abtastrate abhängig. Der Strapdown Algorithmus muss hochfrequent durchlaufen werden[6], jedoch das Aufdatieren des stochastischen Modells in Form der Kovari-

[5] Die Matrizen $\boldsymbol{\Phi}$, $\mathbf{F}$ sowie $\mathbf{G}$ müssen in jeder Epoche k neu berechnet werden.

[6] Manche IMUs liefern bereits vorintegrierte Daten: Geschwindigkeitsinkremente $\delta\mathbf{v}$ und Orientierungsänderungen $\delta\mathbf{q}$. Dann kann auch hier mit niedrigerer Frequenz gearbeitet werden.

anzmatrix kann mit niederer Frequenz erfolgen (jedoch spätestens bis zu dem Zeitpunkt, an dem ein Kalman-Filter Korrekturschritt stattfinden soll). Wendel (2011) gibt als typischen Wert $t = 0,1\,\mathrm{s}$ an. Das hat auch numerische Vorteile, da bei sehr kurzen Zeitschritten und rauscharmen Sensoren die stochastische Modellierung bei 32-Bit IEEE 754 Floating Point Units evtl. nicht mehr darstellbar ist. Für solche Fälle schafft eine Square-Root Formulierung relativ einfach Abhilfe, da hier nicht mit den numerisch ungünstigen Varianzen in der Kovarianzmatrix gearbeitet werden muss, sondern mit deren Wurzeln. Zu dem Thema gibt Kap. 6.5 in Grewal und Andrews (2001) einen guten Überblick[7]. Beispielsweise sei auf den Schmidt-Householder Propagationsalgorithmus in Kombination mit Neil Carlsons Cholesky-basierter Square-Root Formulierung hingewiesen.

4.3 Sensorfusionsgleichungen für verteilte Sensoren

4.3.1 Navka Leverarm- und Plattformkonzept

In Wagner (2003) werden verteilte Sensoren sowie flexible Fahrzeugstrukturen betrachtet, wobei nicht alle Sensoren im Detail analysiert werden und der Fokus eher auf den flexiblen Strukturen liegt. He und Jianye (2002) analysieren die Navigationsfehler, die durch einen nicht modellierten IMU-Leverarm bei hochdynamischen Flugmanövern auftreten und berichten von signifikanten Fehler (bspw. 4,7 m/s Geschwindigkeitsfehler durch einen IMU Leverarmfehler von etwas mehr als 6 m). In Bancroft (2010) wird die Fusion von mehreren verteilten IMUs für ein Fußgängernavigationssystem untersucht, hier werden die Einflüsse ebenfalls als signifikant bezeichnet. In den Standardwerken wird die Verteilung der Sensoren nur zum Teil betrachtet, meistens im Bezug auf den Hebelarm der GNSS Antennen. Folgende Abschnitte tragen die relevanten Zusammenhänge bei verteilten Sensoren in kompakter Form zusammen. Hier wird auf die Arbeiten in dem Verbundforschungsprojekt *Navka*[8] aufgebaut, das als eines der Hauptziele die strenge Behandlung von einer beliebigen Anzahl an verteilten Sensoren und die Zustandsschätzung auf Basis von verteilten Sensoren hat, siehe Abbildung 4.1. Folgende Veröffentlichungen geben einen tieferen Einblick: Jäger et al. (2012), Jäger et al. (2013a), Jäger et al. (2013b) sowie Jäger und Zwiener (2016). Das *Navka*-Konzept gruppiert Sensoren auf verschiedenen verteilten Plattformen (siehe Abbildung 4.1). Bei diesem Konzept erhalten die Sensoren (GNSS, MEMS, FOG, RLG, MOEMS, Einzelsensoren) zum einen den Plattformindex j, unter welchem die Sensorposition und Orientierung auf der j-ten Plattform verwaltet wird und den Sensorindex i. Für jeden Sensor i ist der Bezug zur Plattform j hinterlegt (Orientierung und Leverarm) sowie für jede Plattform j ist der Bezug zum körperfesten Koordinatensystem b eingemessen. Die allgemeine Sensorverortungs- bzw. Leverarmgleichung lautet:

$$\mathbf{x}^n_{s_{ij}} = \mathbf{x}^n_b + \mathbf{R}^n_b \cdot \left[\mathbf{o}^b_{p_j} + \mathbf{R}^b_{p_j} \cdot \mathbf{o}^{p_j}_{s_{ij}} \right]. \tag{4.14}$$

[7] Bei numerischen Problemen in Zusammenhang mit einem Kalman-Filter kann allgemein Grewal und Andrews (2001) als gute Anlaufstelle angesehen werden.

[8] Navka Projekt `www.navka.de` unter der Leitung von Prof. R. Jäger. Von 2011–2013 B.W.-Verbundforschungsprojekt, von 2012–2015 ZIM Projekt *Volocopter*.

Für Softwareimplementierungen ist es sinnvoll, wenn die benötigten Transformationen beim Start vorberechnet werden, sodass zur Laufzeit direkt im körperfesten Koordinatensystem gerechnet werden kann.

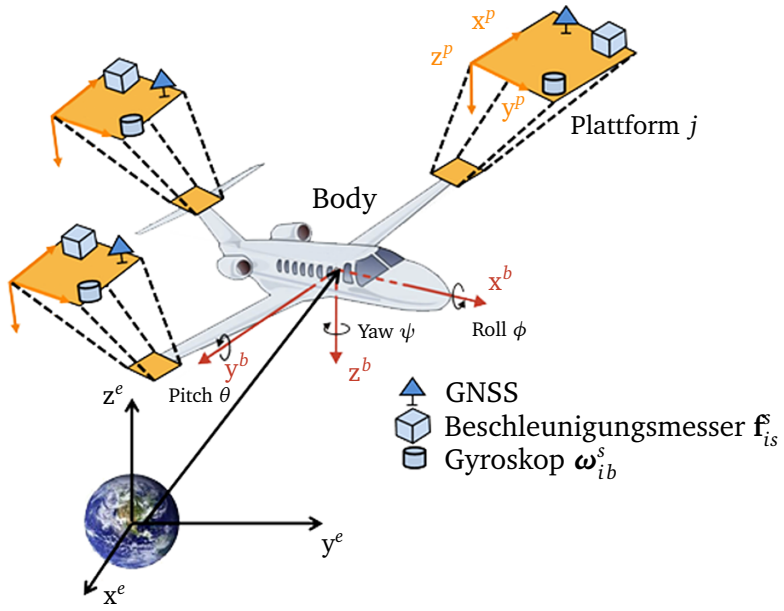


Abbildung 4.1.: Navka Konzept der verteilten Sensoren auf verteilten Plattformen (Jäger et al., 2013b).

4.3.2 GNSS

Die Position einer i-ten GNSS Antenne auf einer j-ten Plattform p im körperfesten Koordinatensystem b lautet allgemein:

$$\mathbf{o}^b_{s_{ij}} = \mathbf{o}^b_{p_j} + \mathbf{R}^b_{p_j} \cdot \mathbf{o}^{p_j}_{s_{ij}}. \tag{4.15}$$

Der Vektor $\mathbf{o}^b_{p_j}$ beschreibt die Position der j-ten Plattform im b-Frame, die Matrix $\mathbf{R}^b_{p_j}$ die Orientierung dieser Plattform bezüglich des körperfesten Systems und $\mathbf{o}^{p_j}_{s_{ij}}$ die Position der i-ten Antenne im Plattform-Koordinatensystem.

Im Folgenden wird der Übersichtlichkeit wegen immer von der i-ten Antenne ausgegangen, auch wird davon ausgegangen, dass der Leverarm bekannt[9] ist und in das körperfeste Koordinatensystem nach Gl. 4.15 vortransformiert wurde. Der Vektor $\mathbf{o}^b_{s_{ij}}$ verkürzt sich im Folgenden also von der Notation her zu: $\mathbf{o}^b$.

[9] Eine Online-Schätzung der Hebelarm-Parameter ist möglich, aber aus Sicherheitsgründen nicht adäquat für ein Luftfahrzeug. Hong et al. (2006) erreichen eine cm-genaue Online-Schätzung des Leverarms. Das von Hong et al. (2006) verwendete Kalman-Filter entspricht dem hier vorgestellten Design.

Die an dieser Antenne gemessene Geschwindigkeit $\dot{\mathbf{x}}^n_{\text{GNSS Ant.}}$ ist abhängig von dem GNSS Antennen Leverarm von Body-Zentrum zu GNSS Antennenphasenzentrum $\mathbf{o}^b$ und der Winkelgeschwindigkeit des Körpers:

$$\mathbf{o}^b = \mathbf{o}^b_{p_j} + \mathbf{R}^b_{p_j} \cdot \mathbf{o}^{p_j}_{s_{ij}} \tag{4.16}$$

$$\dot{\mathbf{x}}^n_{\text{GNSS Ant.}} = \dot{\mathbf{x}}^n + \mathbf{R}^n_b \cdot \left(\boldsymbol{\omega}^b_{nb} \times \mathbf{o}^b\right). \tag{4.17}$$

Diese Leverarm Korrektur wird bei kleinen UAVs oft ignoriert, aber bei hoher Flugdynamik ergeben sich nach Gl. 4.17 durchaus signifikante Beiträge (He und Jianye, 2002).

Zwar ist das vorliegende Filter-Design auf die Integration der rohen GNSS Beobachtungen ausgelegt, aber der Vollständigkeit wegen sollen hier auch die Beobachtungsgleichung für GNSS Geschwindigkeitsmessungen angegeben werden. Das ist die Differenz aus der geschätzten Geschwindigkeit $\hat{\dot{\mathbf{x}}}^n$ gegenüber der gemessenen Geschwindigkeit an der Antenne:

$$\delta \mathbf{l}_{\text{GNSS Vel.}} + \mathbf{v} = \delta \dot{\mathbf{x}}^n = \hat{\dot{\mathbf{x}}}^n + \mathbf{R}^n_b \cdot \left(\boldsymbol{\omega}^b_{nb} \times \mathbf{o}^b\right) - \mathbf{R}^n_e \cdot \dot{\mathbf{x}}^e_{\text{GNSS Ant.}} \tag{4.18}$$

mit der Designmatrix $\mathbf{A}$, nach einer Ableitung von Wendel (2011), bezogen auf den Zustandsvektor (Gl. 4.1)

$$\mathbf{A}_{\text{GNSS Vel.}} = \begin{bmatrix} \mathbf{0}_{3\times3} & \mathbf{I}_{3\times3} & -\left[\mathbf{R}^n_b \cdot \boldsymbol{\Omega}^b_{nb} \cdot \mathbf{o}^b \times\right] & \mathbf{0}_{3\times3} & \mathbf{R}^n_b \cdot \left[\mathbf{o}^b \times\right] & \mathbf{0}_{3\times1} & \mathbf{0}_{3\times1} \end{bmatrix}. \tag{4.19}$$

An der Designmatrix sieht man, dass ein Leverarm direkt die Lage stützt (mittlerer Teil von $\mathbf{A}$) sowie bei der Schätzung des Gyroskop Drehratenbiasfehlers hilft (rechter Teil von $\mathbf{A}$).

GNSS Positionsmessungen können selbstverständlich ebenfalls als Beobachtungen genutzt werden, mit der Differenz aus der geschätzten Position $\hat{\mathbf{x}}^e$ im erdfesten System e gegenüber der an der Antenne gemessenen Position $\mathbf{x}^e_{\text{GNSS Pos.}}$

$$\delta \mathbf{l}_{\text{GNSS Pos.}} + \mathbf{v} = \mathbf{R}^n_e \cdot \left(\hat{\mathbf{x}}^e - \mathbf{x}^e_{\text{GNSS Pos.}}\right) + \mathbf{R}^n_b \cdot \mathbf{o}^b. \tag{4.20}$$

Mit der Designmatrix $\mathbf{A}$ für Positionsbeobachtungen, wieder bezogen auf den Zustandsvektor aus Gl. 4.1:

$$\mathbf{A}_{\text{GNSS Pos.}} = \begin{bmatrix} \mathbf{I}_{3\times3} & \mathbf{0}_{3\times3} & -\left[\mathbf{R}^n_b \cdot \mathbf{o}^b \times\right] & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times1} & \mathbf{0}_{3\times1} \end{bmatrix}. \tag{4.21}$$

Hier ist hervorzuheben, dass die Residuen aus $\delta \mathbf{l}_{\text{GNSS Pos.}}$ nicht nur einen offensichtlichen Beitrag zur Positionierung liefern, sondern auch direkt die Lage stützen.

Für die Pseudorange-Messung muss pro Satellit s die Differenz aus der erwarteten minus der tatsächlich beobachteten Messung PSR^s gebildet werden (vgl. Pseudorange Gl. 3.54):

$$l_{\mathrm{PSR}^s} + v = \delta\mathrm{PSR}^s + v = |\mathbf{n}| + c \cdot \hat{t}_r - c \cdot t^s + \Delta T + \Delta I - \mathrm{PSR}^s. \tag{4.22}$$

Mit dem Hilfsvektor $\mathbf{n}$, der im n-Frame von der Antenne zur Position[10] des Satellits $\mathbf{r}^s$ zeigt (im erdfesten System).

$$\mathbf{n} = \mathbf{R}_e^n \cdot (\mathbf{r}^s - \hat{\mathbf{x}}^e) - \mathbf{R}_b^n \cdot \mathbf{o}^b. \tag{4.23}$$

Die einzeilige Designmatrix $\mathbf{A}$ ist analog zu Gl. 4.21 aufgebaut:

$$\mathbf{e} = \frac{\mathbf{n}}{|\mathbf{n}|} \tag{4.24}$$

$$\mathbf{A}_{\mathrm{PSR}^s} = \begin{bmatrix} -\mathbf{e}^T & \mathbf{0}_{1\times3} & -\mathbf{e}^T \cdot \left[\mathbf{R}_b^n \cdot \mathbf{o}^b \times\right] & \mathbf{0}_{1\times3} & \mathbf{0}_{1\times3} & 1 & 0 \end{bmatrix}. \tag{4.25}$$

4.3.3 Beschleunigungsmesser

Für ein beliebig verortetes Accelerometer im Sensorframe s (implizit auf einer Plattform p) ist folgender Zusammenhang gegeben

$$\begin{aligned} \mathbf{f}^s_{s_{ij}} + \mathbf{v} = & \mathbf{R}_b^s \cdot \mathbf{f}^b + \\ & \mathbf{R}_b^s \cdot \boldsymbol{\Omega}_{ib}^b \cdot \boldsymbol{\Omega}_{ib}^b \cdot \mathbf{o}^b + \\ & \mathbf{R}_b^s \cdot \dot{\boldsymbol{\Omega}}_{ib}^b \cdot \mathbf{o}^b \end{aligned} \tag{4.26}$$

mit

- $\mathbf{f}^s_{s_{ij}}$ Gemessene spezifische Kraft *specific force* des Accelerometers i im Sensor Frame auf der Plattform j
- $\mathbf{f}^b$ Spezifische Kraft (*specific force*) einer (virtuellen) IMU im b-Frame Ursprung
- $\dot{\boldsymbol{\Omega}}_{ib}^b$ Änderungen der Drehraten
- $\mathbf{o}^b$ Leverarm des Sensors im körperfesten Koordinatensystem
- $\boldsymbol{\Omega}_{ib}^b$ Drehraten
- $\mathbf{R}_s^b$ Body Frame zu Sensor Rotation
- $\mathbf{v}$ Messrauschen

[10] Die Satellitenposition muss um den *Sagnac* Effekt korrigiert werden, siehe Gl. 3.59.

Der Leverarm im b-Frame $\mathbf{o}^b$ lässt sich mit Gl. 4.15 bestimmen. Ebenso lässt sich $\mathbf{R}_s^b$ durch die Zusammenhänge in Gl. 4.28 vorberechnen. Abschnitt 4.3.7 behandelt die Bestimmung der Drehratenänderungen.

4.3.4 Gyroskope

Die Beobachtungen eines Gyroskops sind invariant gegenüber Verschiebungen auf der Plattform. Die Orientierung des Gyroskops muss aber berücksichtigt werden, um die Drehraten aus dem Sensorframe s in das körperfeste Koordinatensystem b zu überführen:

$$\mathbf{R}_s^b = \mathbf{R}_p^b \cdot \mathbf{R}_s^p \tag{4.27}$$

$$\boldsymbol{\omega}_{ib}^b + \mathbf{v} = \mathbf{R}_s^b \cdot \boldsymbol{\omega}_{ib}^s. \tag{4.28}$$

An dieser Stelle stellt sich die Frage, wie mehrere verteilte Gyroskope und Accelerometer in die Navigationszustandsschätzung eingebunden werden können. Für ein Total-State Kalman-Filter ist dies vergleichsweise einfach möglich, indem die „virtuelle IMU“ im Ursprung des körperfesten Koordinatensystem im Navigationszustandsvektor $\mathbf{y}$ aufgenommen wird, siehe Jäger und Zwiener (2016), Jäger (2018) oder auch Zwiener (2012). Beispielsweise würde nach Jäger (2018) der Total-State Zustandsvektor $\mathbf{y}$ folgende Größen aufnehmen (im erdfesten System):

$$\mathbf{y} = \begin{pmatrix} \mathbf{x}^e & \dot{\mathbf{x}}^e & \ddot{\mathbf{x}}^e & \boldsymbol{\psi}_b^e & \boldsymbol{\omega}_{eb}^b & \dot{\boldsymbol{\omega}}_{eb}^b \end{pmatrix}^T. \tag{4.29}$$

Da die IMU Beobachtungen in einem Error-State Kalman-Filter über einen Strapdown-Algorithmus (siehe Abschnitt 3.3) verarbeitet werden, ist dieser Ansatz nicht einfach übertragbar. Diese Verfahren benötigen z. T. Messungen und Zuständsschätzungen aus zukünftigen und vergangenen Epochen. Gerade in Zeitabschnitten, wo die Zustandsschätzung nur über IMU Messungen fortgeführt wird (engl. *coasting*).

Ein Multisensorkonzept auf Basis einer virtuellen IMU für die Fahrzeugnavigation wird von Dziubek et al. (2012) vorgestellt. In Ebner und Mark (1978) werden verschiedene IMUs für eine Flugsteuerung über einen einfachen Least-Squares Ansatz verbunden. Pejsa (1974) zeigt, dass Multi-IMUs für eine erhöhte Systemgenauigkeit in einer schiefen Formation angebracht werden sollten. Pittelkau (2005) beschreibt ein Kalibrierverfahren für Multi-IMU Anordnungen. In Waegli et al. (2008) wird ein Multi-IMU System mit GPS Stützung untersucht. Bancroft (2010) betrachtet verschiedene Multi-IMU Verarbeitungsverfahren für ein Fußgängernavigationssystem, und beschreibt ein Virtual IMU (VIMU) Kalman-Filter. Der Zustandsvektor enthält drei Vektoren (ähnlich wie Gl. 4.29):

$$\mathbf{y}_{\text{VIMU}} = \begin{pmatrix} \mathbf{f}^b & \boldsymbol{\omega}_{ib}^b & \dot{\boldsymbol{\omega}}_{ib}^b \end{pmatrix}. \tag{4.30}$$

Aufgrund der hohen Drehratenänderungen von $40000 \frac{^\circ}{s^2}$ von Foot-Mounted-IMUs berichtet Bancroft, dass die Wahl des passenden Prozessrauschens schwierig ist und schlägt eine adaptive Erweiterung für das Kalman-Filter vor, das über ein 500 ms Zeitfenster das Prozessrauschen schätzt. Zudem wird emp-

fohlen, die IMUs auf 0,02 ms genau zu synchronisieren, was durchaus eine praktische Herausfordung darstellt.

4.3.5 Barometrische Höhenmessung

Um eine barometrische Höhenmessung einzubinden, wird zuerst die erwartete barometrische Höhe berechnet: $\hat{h}$. Da das Error-State Kalman-Filter keine Vorschriften macht, wie der *Total-State* $\mathbf{y}$ parametrisiert ist, wird von dieser allgemeinen Formulierung ausgegangen. Wenn z. B. im *Total-State* direkt die barometrische Höhe mitgeführt wird, dann ist die Berechnung von $\hat{h}$ trivial. Wird dagegen beispielsweise die ellipsoidische Höhe in $\mathbf{y}$ geführt, muss diese ellipsoidische Höhe erst in eine barometrische Höhe umgerechnet werden. Meister (2010) schätzt z. B. als *nuisance* Parameter den Offset zwischen der ellipsoidischen Höhe und der barometrischen Höhe zusätzlich in dem Zustandsvektor $\mathbf{y}$ und modelliert die Variationen dieses Barometer-Offsets als Random-Walk. Meister (2010) bindet Barometermessungen mit 120 Hz in die Zustandsschätzung für einen Quadrokopter ein. Unabhängig von der gewählten Strategie lautet die Differenz zwischen der erwarteten barometrischen Höhe $\hat{h}$ und dem tatsächlichen barometrischen Höhenmesswert h in Metern (siehe Abschnitt 3.6):

$$\delta l_{\text{Baro.}} + v = -\left(\hat{h} - h\right) + \begin{pmatrix} -\sin\theta \\ \cos\theta \cdot \sin\phi \\ \cos\theta \cdot \cos\phi \end{pmatrix}^T \cdot \mathbf{o}^b \tag{4.31}$$

oder in Matrix Schreibweise:

$$\mathbf{e} = \begin{pmatrix} 0 & 0 & 1 \end{pmatrix} \tag{4.32}$$

$$\delta l_{\text{Baro.}} + v = -\left(\hat{h} - h\right) + \mathbf{e} \cdot \mathbf{R}_b^n \cdot \mathbf{o}^b. \tag{4.33}$$

Der Vektor $\mathbf{o}^b$ ist der Leverarm des Barometers im körperfesten Koordinatensystem. Die Messdifferenz wird invertiert, da der Zustandsvektor $\mathbf{y}$ in einem *North-East-Down* (NED) System parametrisiert ist. Mit der Designmatrix $\mathbf{A}$ für den Fehler in der barometrischen Höhenmessung:

$$\mathbf{A}_{\text{Baro.}} = \begin{bmatrix} \mathbf{e} & \mathbf{0}_{1\times3} & -\mathbf{e} \cdot \left[\mathbf{R}_b^n \cdot \mathbf{o}^b \times\right] & \mathbf{0}_{1\times3} & \mathbf{0}_{1\times3} & 0 & 0 \end{bmatrix}. \tag{4.34}$$

4.3.6 Magnetometer

Wie in Abschnitt 3.7 beschrieben, lässt sich mit der Beobachtung der magnetischen Flussdichte die Nordrichtung bestimmen. Wie bereits in Abschnitt 4.2 diskutiert benötigt ein auf INS/GNSS aufbauendes Navigationssystem immer wieder Phasen der Beschleunigung, um die Azimutschätzung zu stützen, da diese sonst divergiert. Als Mitigationsstrategie empfehlen Wendel et al. (2006) bzw. Meister (2010) ein Magnetometer über den folgenden Ansatz in ein Error-State Filter zu integrieren. Der Ansatz sieht

vor, dass zwar der volle Drei-Komponenten Vektor als Messung eingeht, aber nur verbessernd auf den Yaw-Winkel einwirkt.

Bei einer realistischen Objektausdehnung können Hebelarmeffekte ignoriert werden. Die Messung eines Magnetometers wird zuerst in das körperfeste Koordinatensystem transformiert:

$$\mathbf{l}^b_{\text{Mag.}} = \mathbf{R}^b_s \cdot \mathbf{l}^s_{\text{Mag.}} \tag{4.35}$$

Das Residuum, das als Beobachtung in die Kalman-Filter Ausgleichung eingeht, wird berechnet aus der Differenz zwischen der gemessenen magnetischen Flussdichte $\mathbf{l}^b_{\text{Mag.}}$ abzüglich der erwarteten mag. Flussdichte[11]

$$\delta\mathbf{l}_{\text{Mag.}} = \mathbf{l}^b_{\text{Mag.}} - \mathbf{R}^b_n \cdot \mathbf{l}^n_{\text{Modell}} \tag{4.36}$$

mit dem Vektor $\mathbf{l}^n_{\text{Modell}}$ aus einem Erdmagnetfeldmodell:

$$\mathbf{l}^n_{\text{Modell}} = f\,(B, L, h, \text{Zeit}) = \begin{pmatrix} m_n \\ m_e \\ m_d \end{pmatrix}. \tag{4.37}$$

Die zugehörige Designmatrix $\mathbf{A}$ ist so aufzubauen, dass nur die den Yaw-Winkelfehler betreffende Spalte besetzt ist:

$$\mathbf{A}_{\text{Mag.}} = \begin{bmatrix} \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times2} & -\mathbf{R}^b_n \cdot \begin{pmatrix} m_e \\ -m_n \\ 0 \end{pmatrix} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & 0 & 0 \end{bmatrix}. \tag{4.38}$$

Der Modellvektor geht sowohl in die Messung als auch in die Designmatrix ein und ist selbst mit Unsicherheiten behaftet, was hier vernachlässigt wird. Aufgrund der allgemeinen Störanfälligkeit von Magnetometermessungen, müssen diese jedoch ohnehin über das reine Sensorrauschen hinaus pessimistischer gewichtet werden (Filter-Tuning). Robuste Schätzmethoden bieten sich an. Meister (2010) nutzt 64 Hz als Magnetometermessfrequenz; für die beschriebenen Testträger in dieser Arbeit haben sich auch 50 Hz als ausreichend erwiesen.

4.3.7 Änderungen der Drehraten für verteilte Sensoren

Um die Messungen in einem System mit verteilten Sensoren richtig verarbeiten zu können, ist die Änderung der Drehrate notwendig ($\dot{\boldsymbol{\omega}}^b_{ib}$), siehe Gleichung 4.26.

[11] Üblicherweise über das *World Magnetic Model* (Chulliat et al., 2015). In Mitteleuropa führt ein Vernachlässigen des Modellvektors zu Fehlern zwischen $0-10°$ im Heading, weltweit bis zu 90°.

Da dafür kein Sensor existiert muss die Größe aus den Gyroskopmessungen abgeleitet werden. Eine einfache numerische Differentiation (Gleichung 4.39) über den Differenzenquotienten ist naheliegend, verstärkt aber das vorhandene Messrauschen des Gyroskops bzw. kann zu komplett falschen Ergebnissen führen.

$$\dot{\boldsymbol{\omega}} = \frac{\boldsymbol{\omega}(t+\Delta t) - \boldsymbol{\omega}(t)}{\Delta t} \tag{4.39}$$

Ein einfaches Glätten der Gyroskopmessungen ist möglich; dadurch werden aber Latenzen eingeführt, was wieder zu Verfälschungen führt. In Jäger (2018) wird ein Total-State Zustandsvektor vorgeschlagen, der den Navigationszustandsvektor $\mathbf{y}$ erweitert und die Änderungen der Drehraten aufnimmt. Bancroft (2010) behandelt dieses Thema ebenfalls und schlägt ein separates Kalman-Filter vor, das die Drehraten und die Drehratenänderungen schätzt (siehe Gl. 4.30). Diese Methode ist prinzipiell möglich, hat aber den Nachteil, dass das Filter stochastisch richtig modelliert sein muss. Ein zu hoher oder zu niedriger Rauschenanteil kann das Ergebnis negativ beeinflussen und zu Verzögerungszeiten führen. Daher soll im Folgenden ein Ansatz gezeigt werden, der auch ohne Latenzen zu einer Schätzung kommt, aber dennoch glättend wirkt.

Abschnittsweise wird ein Ausgleichspolynom auf die neuesten Drehraten gelegt. Von diesem Ausgleichspolynom kann dann die erste Ableitung ($\frac{d\omega}{dt}$) bestimmt werden. Dieser Ansatz kommt mit verrauschten Daten zurecht und führt gleichzeitig keine Latenzen ein, wenn das Polynom auf dem neuesten Datenpunkt ausgewertet wird. Das Verfahren wurde von Abraham Savitzky und Marcel J. E. Golay in einem populären Beitrag veröffentlicht (Savitzky und Golay, 1964), alternativ wird die Methode auch *least-squares filter* oder DISPO (Digital Smoothing Polynomial Filter) genannt. Das Besondere an dem Verfahren von Savitzky und Golay (1964) gegenüber einem gewöhnlichen Least-Squares Polynomial Fitting ist, dass man zur Laufzeit keine komplette Ausgleichungsrechnung durchführen muss, sondern die Ausgleichung schon als Filter Koeffizienten vorberechnen kann. Somit muss nicht für jeden Abtastpunkt ein Normalgleichungssystem gelöst werden, was den Rechenaufwand stark verkürzt. Was jedoch voraussetzt, dass die Abtastintervalle gleich sind. Davon kann aber bei Gyroskopen ausgegangen werden.

Durch gegebene Funktionswerte bzw. Messungen $f_{-n_L}, \ldots, f_{n_R}$ wird ein Polynom $a_0 + a_1 \cdot i + \ldots + a_M \cdot i^M$ vom Grad M an den Stellen $i = -n_L \ldots n_R$ mit $i \in \mathbb{Z}$ gelegt. Daraus ergibt sich, wie für Polynomausgleichungsprobleme üblich, eine VANDERMONDE[12] Designmatrix $\mathbf{A}$: $\mathbf{A}_{ij} = i^j$ mit $j = 0, \ldots, M$. Gesucht ist der Vektor der Polynomkoeffizienten $\mathbf{a}$.

$$\mathbf{A} \cdot \mathbf{a} = \mathbf{f} \tag{4.40}$$

$$\mathbf{a} = \left(\mathbf{A}^T \cdot \mathbf{A}\right)^{-1} \cdot \mathbf{A}^T \cdot \mathbf{f} \tag{4.41}$$

[12] nach Alexandre-Théophile Vandermonde

Beispielsweise für ein Polynom zweiter Ordnung ($M = 2$) und den Stützstellen $n_L = 2$ und $n_R = 2$ sieht die VANDERMONDE Matrix **A** folgendermaßen aus:

$$\mathbf{A} = \begin{pmatrix} 1 & -2 & (-2)^2 \\ 1 & -1 & (-1)^2 \\ 1 & 0 & (0)^2 \\ 1 & 1 & (1)^2 \\ 1 & 2 & (2)^2 \end{pmatrix} \tag{4.42}$$

mit dem Polynom:

$$\mathbf{a} = (a_0, a_1, a_2)^T \tag{4.43}$$

$$f_i = a_0 + a_1 \cdot i + a_2 \cdot i^2 \tag{4.44}$$

und $i = \{-2, -1, 0, 1, 2\}$.

Entscheidend ist nun bei dem S-G Filter, dass gar nicht der Vektor der Polynomkoeffizienten **a** gesucht wird, sondern ein Satz von Koeffizienten **c**, die mit den verrauschten Funktionswerten f_i gefaltet werden und so jeweils einen geglätteten Funktionswert g_i ergeben. Also in einem Sliding-Window Verfahren wird der Vektor **c** über die verrauschten Eingangsdaten f_i gelegt:

$$g_i = \sum_{n=-n_L}^{n_R} c_n \cdot f_{i+n}. \tag{4.45}$$

In Press et al. (2007) wird gezeigt, dass der Vektor **c** der ersten Zeile der Matrix **K** entspricht[13]:

$$\mathbf{K} = \left(\mathbf{A}^T \cdot \mathbf{A}\right)^{-1} \cdot \mathbf{A}^T. \tag{4.46}$$

Die zweite Zeile in Matrix **K** aus Gleichung 4.46 entspricht einem Satz an Koeffizienten, die der ersten Ableitung entsprechen. Allerdings muss das Ergebnis der Faltung noch durch die Abtastzeit Δt dividiert werden. In Anhang C ist ein Programm angehängt, das die beschriebenen Berechnungen durchführt. Z. B. für ein Polynom zweiter Ordnung sind Parameter in Tabelle 4.2 aufgelistet, die direkt als Koeffizienten in Gleichung 4.45 genutzt werden können. Die Parameter in der zweiten und dritten Zeile (für $n_R = 0$) sind interessant, da hier keine Pufferung der Eingangsdaten vorgenommen werden muss und somit keine Latenzen entstehen. Falls es die Anwendung erlaubt, ist aber eine Pufferung dennoch empfehlenswert, da sich die Genauigkeit durch den Blick in Zukunft i. A. deutlich erhöht. Verschiedene Sätze an Koeffizienten für die erste Ableitung sind in Tabelle 4.3 gegeben. Press et al. (2007) empfehlen für die Bestimmung der ersten Ableitung Polynome 4. Ordnung. Sowie allgemein als Faustregel für das Smoo-

[13] Gleichung 4.46 kann um eine Gewichtsmatrix **P** erweitert werden, falls in speziellen Anwendungsfällen bestimmte Messwerte einen besonderen Einfluss bekommen sollen.

Tabelle 4.2.: Savitzky-Golay Koeffizienten zur Glättung.

M	n_L	n_R	-6	-5	-4	-3	-2	-1	0	1	2
2	2	2					-0.086	0.343	0.486	0.343	-0.086
2	4	0			0.086	-0.143	-0.086	0.257	0.886		
2	6	0	0.119	-0.071	-0.143	-0.095	0.071	0.357	0.762		
2	6	2	-0.103	-0.009	0.068	0.127	0.169	0.194	0.201	0.191	0.164

Tabelle 4.3.: Savitzky-Golay Koeffizienten f. die geglättete Ableitung 1. Ordnung.

M	n_L	n_R	-6	-5	-4	-3	-2	-1	0	1	2
2	2	2					-0.200	-0.100	0.000	0.100	0.200
2	4	0			0.371	-0.386	-0.571	-0.186	0.771		
2	6	0	0.250	-0.071	-0.250	-0.286	-0.179	0.071	0.464		
4	6	2	-0.085	0.188	0.042	-0.177	-0.263	-0.147	0.105	0.285	0.053

thing sollten n_L und n_R etwa so breit sein wie die Halbwertsbreite[14] der kleinsten gesuchten Merkmale im Signal. Für umfangreiche Koeffiziententabellen sei auf den Anhang der Originalveröffentlichung von Savitzky und Golay (1964) hingewiesen oder den Programmcode zur Koeffizientenberechnung in Anhang C. Abbildung 4.2 zeigt beispielhaft die Vorteile der Filterungsmethode. Abbildung 4.3 zeigt, wie

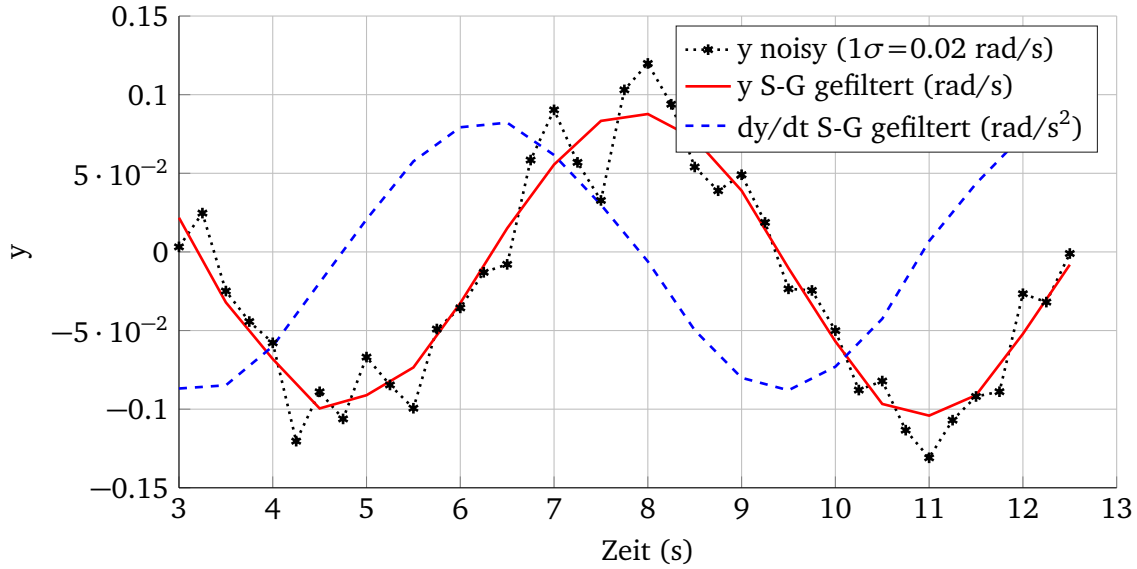


Abbildung 4.2.: Beispiel für die Savitzky-Golay Filterung. Die erste Ableitung kann mit hoher Qualität rekonstruiert werden kann. $M = 4$, $n_L = 12, n_R = 12$.

sich ein Rauschen in Drehraten auf die Bestimmung der Änderungen der Drehraten auswirkt. Dazu wird zum einen aus einem rauschfreien Gyroskop-Signal die erste Ableitung berechnet, dem wird gegenübergestellt, wie sich die erste Ableitung verhält, wenn das Gyroskop-Signal mit einem Rauschen (hier 0,009

[14] engl. full width at half of maximum: FWHM

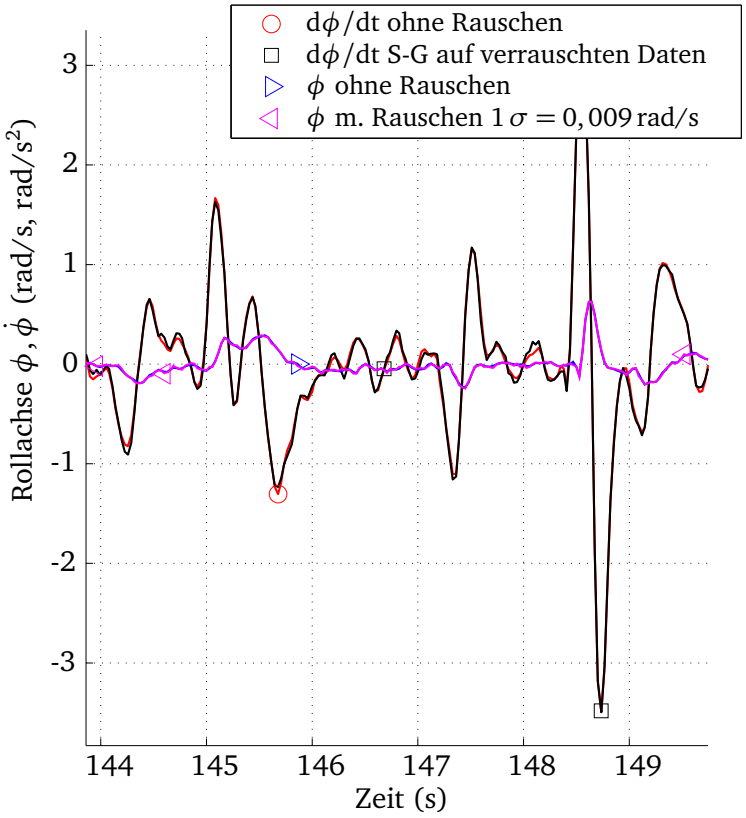


Abbildung 4.3.: Ein 9 Punkt Savitzky-Golay (S-G) Filter kann trotz verrauschtem Gyrokop Eingangssignal die tatsächliche Drehrate approximieren.

rad/s) behaftet ist. Die Erkenntnis ist, dass es kaum einen Unterschied gibt in den beiden berechneten Kurven für $\frac{d\omega}{dt}$, ob mit oder ohne Rauschen.

4.4 Einbeziehung von Messdaten mit verzögerter Verfügbarkeit

Die falsche zeitliche Verarbeitung von Messdaten (gegenüber der IMU) kann die Genauigkeit der Zustandsschätzung signifikant verschlechtern. Beispielsweise geben Van Der Merwe und Wan (2004) an, dass die richtige Verarbeitung von verzögert verfügbaren GNSS Messungen wichtiger ist als die Kalman-Filter Art (Sigma-Point Kalman-Filter vs. EKF). Die rechnerisch an sich beste Strategie ist es alle Sensordaten zu puffern und bei verzögertem Eintreffen der Messdaten wieder alle Berechnungen bis zum aktuellen Zeitpunkt neu zu rechnen. Da dies mit einigem Aufwand verbunden ist, schlägt Wendel (2011) eine Näherungslösung vor, bei der nur eine Historie der Kalman-Filter Zustände $\mathbf{y}_{k-n}$ sowie der zugehörigen Kovarianzmatrizen $\mathbf{Q}_{yy,k-n}$ gespeichert wird. Die verzögert verfügbaren Messungen werden dann mit dem zeitlich am ehesten passenden Zustandsvektor/Kovarianzmatrix-Paar aus der Vergangenheit fusioniert (siehe Gl. 2.103). Die Verbesserungen werden dann aber an den aktuellen Systemzustand angebracht. Puls (2011) verwendet dieses Verfahren um Latenzen von 600 ms zu kompensieren. Das Kompensationsverfahren wurde auch im Rahmen dieser Arbeit verwendet um die verzögerte Verfügbarkeit von GNSS Messungen zu behandeln. Im Folgenden liegt der Fokus auf GNSS Sensoren, da hier große Latenzen auftreten. Das Konzept ist aber übertragbar auf andere Sensoren, wie z. B. Kameras. Barome-

ter hingegen haben eine Latenz von nur wenigen Millisekunden, sodass sich hier der Mehraufwand im Hinblick auf das Rauschverhalten nicht lohnend ist.

Für die Korrektur der Latenz in den GNSS Messungen muss zuerst die tatsächliche Latenz bestimmt werden. Idealerweise bietet der GNSS Empfänger hardwareseitig eine Zeitimpulsoption an. Der GNSS Empfänger setzt eine Ausgangsleitung zur vollen Sekunde der GPS Zeit für eine bestimmte Zeitperiode auf logisch 1 (typischerweise für wenige Millisekunden). Ein Mikrocontroller kann auf die positive Flanke des Signals mit einer Interrupt Funktion reagieren und hat somit einmal pro Sekunde sowohl die lokale Systemzeit (von einem Quartz generiert) und die GPS Zeit (als Time-Of-Week und GPS Woche). Vergleichbar mit einer Zeitansage am Telefon sendet der Empfänger vorab digital die für die nächste positive Flanke gültige GPS Zeit. Mit dieser genau bekannten GPS Zeit kann der Empfänger dann berechnen, wie alt die ankommenden GNSS Messungen sind. Die Latenz schwankt bei einem *M8T* Empfänger der Firma u-blox typischerweise in einem Bereich zwischen 60-120 Millisekunden. Abhängig hauptsächlich von der Satellitenkonstellation und Empfängerkonfiguration.

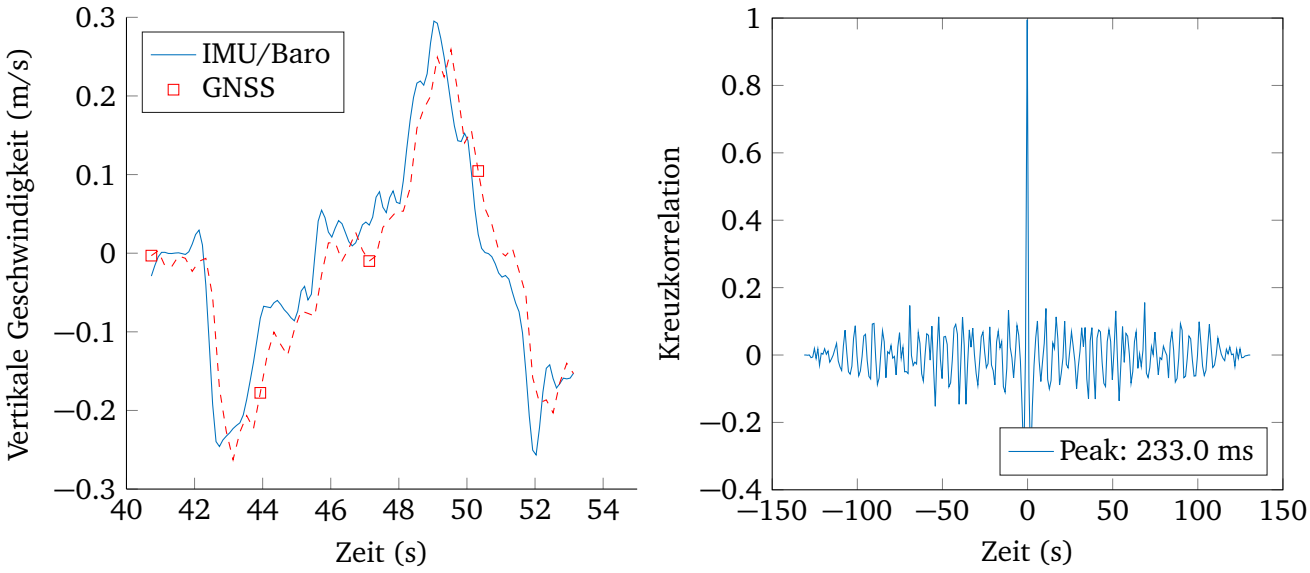
Während das Verfahren für rohe Beobachtungen streng gültig ist, muss bei einer Integration einer vorberechneten Lösung (Position u. Geschwindigkeit) beachtet werden, dass die GNSS Receiver üblicherweise intern selbst ein aktives Kalman-Filter haben. Diese digitale Filterung der Positon u. Geschwindigkeit führt zu weiteren signifikanten Verzögerungen, die mit dem bisher beschriebenen Ansatz nicht korrigiert werden.

Zur Bestimmung der Latenz wird folgende Grundidee genutzt: mit einer IMU/Barometer Sensorkombination lässt sich eine Vertikalgeschwindigkeit praktisch verzögerungsfrei bestimmen (wenige Millisekunden, je nach Tiefpassfilter der Sensoren). Der Zustandsvektor enthält dann (vergleichbar mit Gl. 4.1) u. a. die Vertikalgeschwindigkeit. Diese kann der gemessenen Vertikalgeschwindigkeit des GNSS Empfängers gegenübergestellt werden, so wie in Abbildung 4.4a. Hier sieht man direkt, dass der GNSS Empfänger praktisch die gleiche Vertikalgeschwindigkeit berechnet, allerdings zeitverzögert. Die gemeinsame Zeitbasis ist die lokale Uhr des Navigationscomputers. Mit Hilfe einer Kreuzkorrelation kann der zeitliche Versatz einfach berechnet werden. Abbildung 4.4b zeigt die Korrelation der beiden Signale in Abhängigkeit der Verschiebung. Die größte Korrelation findet sich bei 233 ms. In diesem Beispiel wurde der u-blox M8T Empfänger in dem sogenannten „*Airborne 1G*“ Filtermodus betrieben. Andere Modi verursachen bei diesem Empfänger z. T. unterschiedliche Latenzen.

Für Systeme bzw. Sensoren, die nicht direkt mit dem Navigationscomputer verbunden sind, muss u. U. die genaue Uhrzeit zwischen beiden Systemen bekannt sein. Der Zeitstempel von zwei verschiedenen Uhren lässt sich folgendermaßen in Zusammenhang bringen:

$$t_{\text{System 1}} = (1 + a) \cdot t_{\text{System 2}} + b. \tag{4.47}$$

Der Parameter a wird als Gangabweichung (*engl. clock skew*) bezeichnet. Der Offset ist mit b abgekürzt. Bei geringen Genauigkeitsanforderungen reicht es aus, einmal pro Sekunde den b Term zu berechnen. Bei höheren Ansprüchen kann über ein Ausgleichungsmodell der Parameter a und b bestimmt werden. Ein allgemeinerer Ansatz zur Uhrensynchronisierung von verteilten Systemen, also Systeme die über ein Netzwerk miteinander verbunden sind, wird von Harrison und Newman (2011) unter dem Namen *TICSync* vorgeschlagen. Der Algorithmus basiert auf der Annäherung der Zielfunktion (4.47) durch eine



(a) Vergleich der Vertikalgeschwindigkeiten.

(b) Bestimmung der Latenz über Kreuzkorrelation.

Abbildung 4.4.: Bestimmung der GNSS Messlatenz am Beispiel des u-blox M8T Empfängers.

obere und untere Ausgleichungsgerade und ist sehr effizient. Der TICSync Algorithmus ist quelloffen und benötigt paarweise Uhrzeitmessungen von System 1 und System 2 um dann fortlaufend a und b zu bestimmen. Aber auch auf nicht-verteilten Systemen kann der TICSync Algorithmus genutzt werden. Z. B. kann die Gangabweichung der Uhr des Navigationscomputers bestimmt werden. Abbildung 4.5 zeigt die Bestimmung des Uhrenfehlers eines *STM32F407* Microcontrollers bezüglich der GPS Receiver Zeit (bestimmt durch einen u-blox M8T GNSS Empfänger, wie am Anfang des Abschnitts beschrieben). Nach weniger als 20 Sekunden sind die beiden Uhren synchronisiert und der Uhrenfehler bestimmt.

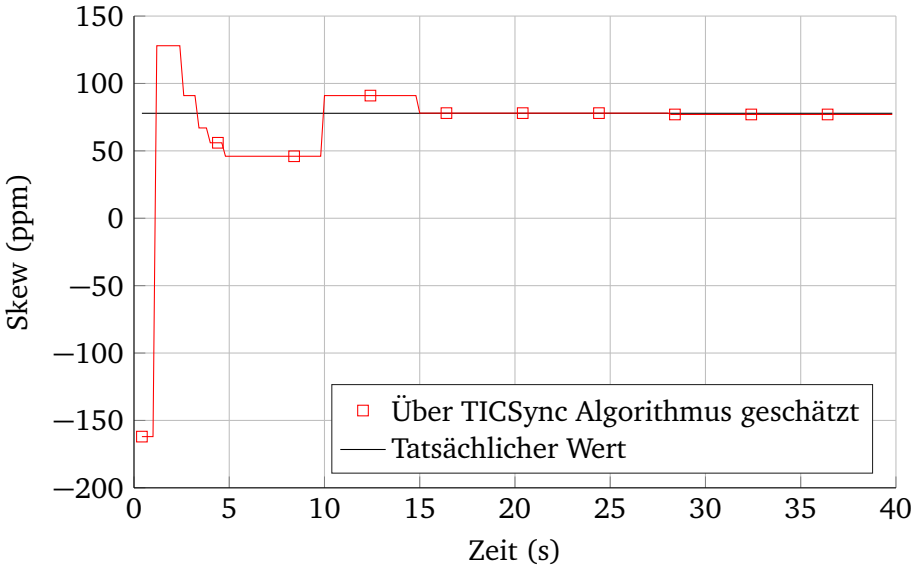


Abbildung 4.5.: Schätzung der Gangabweichung über den TICSync Algorithmus.

4.5 Reduzierte Zustandsschätzung

4.5.1 Lageschätzung

Ein Sonderfall der reduzierten Zustandsschätzung ist ein sogenanntes Attitude und Heading Reference System (AHRS). Ein AHRS bestimmt die Lage eines Körpers, nicht aber die Position oder Geschwindigkeit. Ausgehend von dem kompletten Navigationszustandsvektor (Gl. 4.1) reduziert sich der Zustandsvektor bei einem AHRS signifikant (Gl. 4.48), da keine Positionen und Geschwindigkeiten geschätzt werden sollen, bzw. deren Schätzung nicht möglich ist. Somit ist auch keine Bestimmung des Accelerometer-Offsetfehlers möglich (Wendel, 2011). Der Zustandsvektor reduziert sich auf die Lagefehler sowie die Drehratenbiasfehler:

$$\delta \mathbf{y} = \begin{pmatrix} \delta \boldsymbol{\psi} \\ \delta \mathbf{b}_{\omega} \end{pmatrix}. \tag{4.48}$$

Es gelten die gleichen Vorhersage- und Beobachtungsgleichungen aus Abschnitt 4.3, allerdings müssen in den Matrizen die Elemente gestrichen werden, die keinen Bezug mehr zu dem reduzierten Zustandsvektor haben. Eine ausführliche Herleitung wurde bereits für das linearisierte Kalman-Filter in Abschnitt 2.9.2 betrachtet. Für das AHRS muss zudem die Lage durch Accelerometermessungen gestützt werden. Die Annahme ist, dass sich das zu navigierende Objekt nicht ständig in einem beschleunigten Zustand befindet[15]. Somit entspricht im Mittel die Accelerometermessung der Schwerebeschleunigungsmessung mit umgekehrtem Vorzeichen und kann zur Bestimmung von Roll- und Pitch genutzt werden, siehe Abschnitt 3.2. In Farrell (2008) wird ein Algorithmus vorgestellt und analysiert, der Beschleunigungsphasen erkennt. In diesen Phasen darf die Accelerometermessung nicht zur Lagestützung genutzt werden. Ein robustes Kalman-Filter kann diese Verwerfung auch durchführen. Eine Art Vorfilter Heuristik ist aber in jedem Fall empfehlenswert; nur wenn die Beobachtungen in der Nähe von 9,81 m/s² liegen, sollten diese zur Lagestützung genutzt werden.

Die Beobachtungsgleichung wird gebildet aus der kalibrierten Accelerometermessung abzüglich der erwarteten Schwerebeschleunigung im körperfesten Koordinatensystem (Meister, 2010):

$$\delta \mathbf{l}_{\text{Gravity}} + \mathbf{v} = \mathbf{f}^b - \hat{\mathbf{R}}_n^b \cdot \mathbf{g}^n_{\text{Modell}}. \tag{4.49}$$

Mit der zugehörigen Designmatrix bezogen auf den Zustandsvektor in Gl. 4.48:

$$\mathbf{A}_{\text{Acc.}} = \begin{bmatrix} \mathbf{R}_n^b \cdot \begin{pmatrix} 0 & -g & 0 \\ g & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} & \mathbf{0}_{3\times 3} \end{bmatrix}. \tag{4.50}$$

[15] Für ein Luftfahrzeug ist es aber durchaus denkbar, dass es sich für lange Zeit auf einer Kreisbahn befindet und die gemessenen Kräfte verfälscht sind, was zu einem Lagefehler führt.

Auf eine Stützung des Gierwinkels kann bei Low-Cost Sensorik in der Praxis nicht verzichtet werden. Daher ist eine periodische Messung der magnetischen Flussdichte nach Gl. 4.37 sowie Gl. 4.38 notwendig.

Falls darüber hinaus auch das Azimut nicht geschätzt werden soll (beispielsweise, weil keine Magnotometer zur Stützung verfügbar oder gewünscht sind), reduziert sich der Vektor zu einem 5×1 Vektor zu:

$$\delta \mathbf{y} = \begin{pmatrix} \delta\phi \\ \delta\theta \\ \delta \mathbf{b}_\omega \end{pmatrix}. \tag{4.51}$$

Dieser Zustandsvektor erlaubt eine einfache und vergleichsweise robuste Schätzung der Roll- und Pitchwinkel, verlässt sich aber beim Nordwinkel auf die Integration der Drehraten ohne externe Korrektur. Somit müssen die Gyroskope eine gewisse Güte aufweisen. Je nach Lage kann der Bias der Gyroskop Z-Achse durch Gl. 4.50 gestützt werden, dies setzt allerdings eine gewisse Objektdynamik voraus. Zero Rotation Updates (siehe Abschnitt 3.9) sind für dieses Filter sehr hilfreich um den Gyroskop Bias möglichst früh zu bestimmen.

4.5.2 Höhenbestimmung

Ein Sonderfall der reduzierten Zustandsschätzung ist ein Kalman-Filter zur Höhenbestimmung. Das Modell wurde von Meister (2010) veröffentlicht sowie in Grundzügen in Wendel et al. (2006) beschrieben. Es wird davon ausgegangen, dass eine Schätzung der Lage vorhanden ist. Damit kann zuerst die aktuelle vertikale Beschleunigung $\ddot{x}_d$ berechnet werden:

$$\ddot{x}_d + v = \begin{pmatrix} 0 & 0 & 1 \end{pmatrix} \cdot \left(\mathbf{R}_b^n \cdot \hat{\mathbf{f}}_{ib}^b + \mathbf{g}^n \right). \tag{4.52}$$

Der reduzierte Zustandsvektor besteht aus drei Größen, geschätzt entlang der Down-Achse eines NED Systems:

$$\mathbf{y}_{\text{Baro}} = \begin{pmatrix} x_d^n \\ \dot{x}_d^n \\ b_d^n \end{pmatrix}. \tag{4.53}$$

Die Komponente x_d^n beschreibt die Höhe, $\dot{x}_d^n$ die Vertikalgeschwindigkeit und b_z^n den Biasfehler entlang der Down-Achse. b_d^n ist nicht mit dem Bias in der Accelerometermessung zu verwechseln. Die Größe b_d^n besteht zum Teil aus nicht-kalibrierten Biasfehlern (je nach Lage), sowie aus Lagefehlern, die zu einer fehlerhaften Kompensation der Schwerebeschleunigung führen und Fehlern im Schwerebeschleunigungsmodell selbst. Die Trennung zwischen der Schwerebeschleunigung und Lagefehlern ist schwierig, hier sei für eine tiefergehende Betrachtung auf Becker (2016) verwiesen. Diese Zustandsvariable kann sich also z. B. durch Lageänderungen relativ schnell ändern, daher muss sie durch ein entsprechend ho-

hes Systemrauschen propagiert werden. Die Vorhersage kann direkt im zeitdiskreten Bereich modelliert werden, die gemessene Beschleunigung (Gl. 4.52) wird als Systemeingabe genutzt:

$$\mathbf{y}_{\text{Baro},t+\Delta t} = \underbrace{\begin{pmatrix} 1 & \Delta t & \frac{1}{2}\Delta t^2 \\ 0 & 1 & \Delta t \\ 0 & 0 & 1 \end{pmatrix}}_{\mathbf{\Phi}} \cdot \mathbf{y}_{\text{Baro},t} + \underbrace{\begin{pmatrix} \frac{1}{2}\Delta t^2 \\ \Delta t \\ 0 \end{pmatrix}}_{\mathbf{B}} \cdot \ddot{x}_d + \mathbf{w} \tag{4.54}$$

mit dem stochastischen Vektor $\mathbf{w} \sim \mathcal{N}(0, \mathbf{Q})$, der die Unsicherheit in der Vorhersage modelliert. Üblicherweise wird die barometrische Höhe relativ zum Startpunkt[16] mit dem Barometer gemessen (umgerechnet in Metern, siehe Abschnitt 3.6), da hier automatisch gemeinsame Fehler eliminiert werden:

$$l_{\text{Baro}} = h_{\text{Baro}} - h_{\text{Start}}. \tag{4.55}$$

Die Designmatrix $\mathbf{A}$ ist einfach aufzubauen, da die relative barometrische Höhe direkt als Messung genutzt wird:

$$l_{\text{Baro}} + v = \begin{bmatrix} -1 & 0 & 0 \end{bmatrix} \cdot \mathbf{y}_{\text{Baro},t}. \tag{4.56}$$

4.6 Modellierung von Vibrationen

In Abschnitt 2.10 werden die Grundlagen für die stochastische Modellierung von Sensorfehlern betrachtet. Hinzu kommt allerdings, dass IMUs unter Vibrationen ihre Charakeristik ändern können. Für die

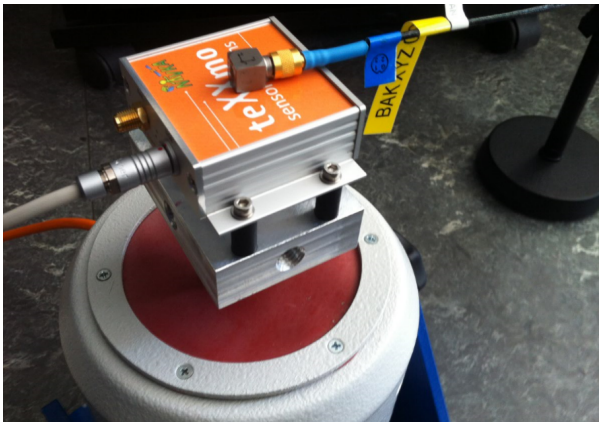
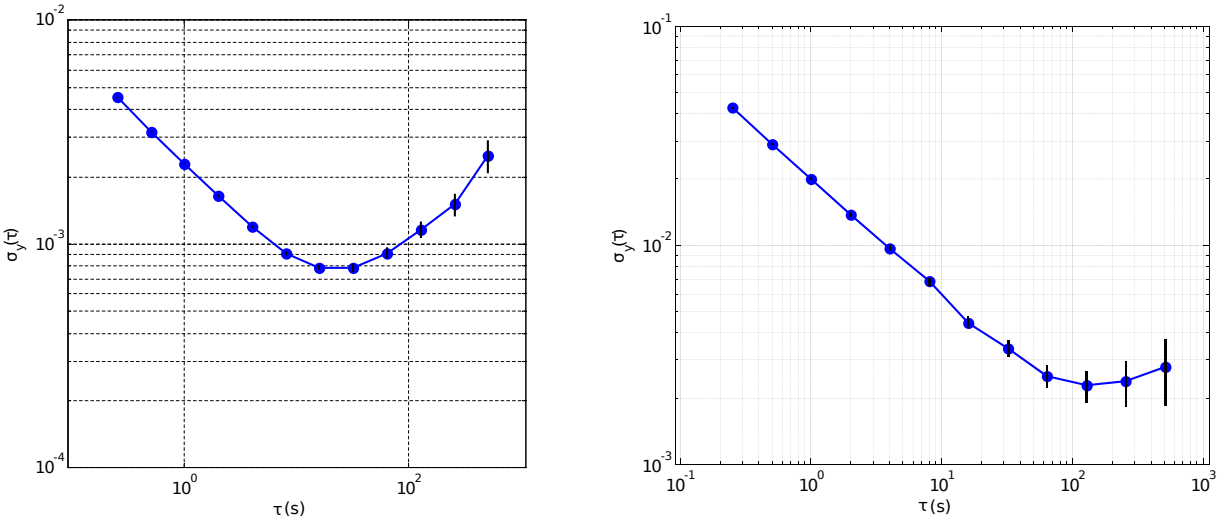


Abbildung 4.6.: Test von IMU auf Shaker[17].

Bestimmung der entsprechenden Parameter (Rauschverhalten, Bias Stabilität, Bias Random Walk) eignet sich z. B. ein Schwingtisch, auch Shaker oder Rütteltisch genannt, mit dem die IMU gezielten Vibrationen unter wiederholbaren Bedingungen ausgesetzt ist. Das ist besonders für Luftfahrtgeräte wichtig.

[16] Zum Startzeitpunkt werden, wenn das Barometer seine Temperaturstabilität erreicht hat, für einige Zeit Messungen gesammelt. Der sich daraus ergebende Mittelwert ist dann als Starthöhe h_{Start} definiert.

Wenn ein Vibrationsprofil des Gefährts vorhanden ist, kann damit der Shaker programmiert werden. Wenn dies nicht bekannt ist, kann eine Vorabschätzung über die typischen Propellerdrehzahlen (RPM) erfolgen. In Abbildung 4.6 ist eine IMU zur Datenaufnahme auf einem Rütteltisch angebracht. Zusätzlich ist ein Beschleunigungsaufnehmer angebracht, was aber für die Bestimmung der Sensorfehler nach Abschnitt 2.10 nicht notwendig ist. Diese werden aus den rohen Beobachtungen der zu prüfenden IMU berechnet. Exemplarisch sollen an dieser Stelle die Parameter eines Low-Cost Accelerometers der Firma



(a) Motoren aus.

(b) Motoren an.

Abbildung 4.7.: Allan Deviation für InvenSense MPU9150 IMU (die eingezeichneten 1-σ Fehlerbalken wachsen mit zunehmender Abschnittslänge).

Invensense vom Typ *MPU-9150* betrachtet werden. Diese IMU wurde untersucht, um die passenden Parameter für die stochastische Modellierung (siehe Abschnitt 4.2) der IMU Fehler bei laufenden Motoren zu bestimmen. In diesem Beispiel wurden die Parameter für ein UAV[18] mit sechs Motoren bestimmt. Für diese Untersuchung wurde die IMU auf dem Fluggerät angebracht. Das Fluggerät wurde fest mit dem Untergrund verbunden und über einen längeren Zeitraum mit konstanten Drehzahlen der Aktorik betrieben um ein möglichst realistisches Vibrationsprofil aufzunehmen. Die IMU wurde in beiden Fäl-

Tabelle 4.4.: Parameter für Low-Cost IMU: Motoren an/aus.

Parameter	Einheit	Motoren aus	Motoren an
Wurzel der spektralen Leistungsdichte	$\frac{m/s^2}{\sqrt{Hz}}$	0.002289	0.020077
Bias Stabilität	m/s^2	0.00078	0.0023
Bias Random Walk	$\frac{m/s^2}{\sqrt{s}}$	0.0001	0.0002

len so konfiguriert, dass ein 42 Hz Tiefpassfilter aktiv war (DLPF). Wenn ein Sensorfehlverhalten durch bestimmte Anregungsfrequenzen ausgelöst wird, dann reicht ein Tiefpassfilter u. U. nicht aus. In dieser Messung wurde die IMU auf Unterlagen vom Typ *Kyosho Z8006 Zeal Gel Pads* angebracht.

[18] Siehe Abbildung 1.5c, Antrieb: 10×45 Propeller, T-Motor MN3110-26, Carbon Rahmen, 6S Spannungsversorgung.

In den Ergebnissen (Tab. 4.4) ist deutlich sichtbar, dass sich die Parameter bei aktiver Aktorik verschlechtern, wohingegen sich der Bias Random Walk beispielsweise verdoppelt. Wird ein derartiges Verhalten nicht modelliert, ist die Zustandsschätzung bei Tests am Boden zwar richtig, aber im Flug bzw. unter Vibrationen fehlerhaft eingestellt, was u. a. zu kritischen Lagefehlern führen kann. Meister (2010) empfiehlt zudem für Multirotor-Fluggeräte aufgrund der Vibrationen eine Filterung der Accelerometermessungen mit einem Filter mit unendlicher Impulsantwort (IIR) 3. Ordnung und geringer Latenz.

4.7 Zustandsautomat für die Navigationszustandsschätzung

Die in Abschnitt 4.5 vorgestellten Zustandsschätzungssysteme (AHRS und Höhenbestimmung) ergänzen sich mit dem allgemeinen Navigationssystem, das in Abschnitt 4.1 beschrieben ist. Das Grundprinzip

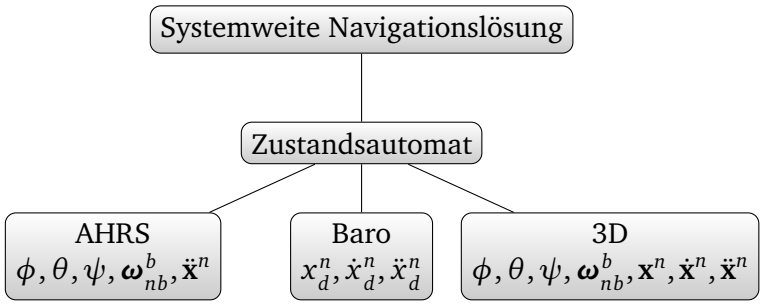


Abbildung 4.8.: Parallele Zustandsschätzer.

dazu wurde von Meister (2010) vorgestellt. Abbildung 4.8 zeigt welche Zustandsvariablen (ohne *nuisance* Parameter) geschätzt werden, dabei ist nicht garantiert, dass jedes Filter immer verfügbar ist. Ein Zustandsautomat muss nun entscheiden, welche Navigationslösung er an den Rest des Systems (z. B. eine Flugsteuerung) weiterleitet. Dieser Zustandsautomat ist in Abbildung 4.9 dargestellt. Mit „*3D*“ ist das allgemeine Filter aus Abschnitt 4.1 gemeint. „*AHRS*“ beschreibt das Lagefilter aus Abschnitt 4.5.1 und „*Baro*“ das Filter zur Höhenbestimmung aus Abschnitt 4.5.2. Ein *Baro Fehler* oder *GNSS Fehler* wäre z. B. ein grundlegender Sensorfehler (Bus-Lesefehler), ein Sprung in den Zeitstempeln, eine zu geringe oder auch unerwartet hohe Abtastrate, offensichtlich ungültige Messwerte oder bei aktivem Filter: ein Fehler in der Kovarianzmatrix. Meister (2010) hat noch weiterführende Kriterien aufgeführt, die in der Praxis einen Zustandswechsel erfordern. Ein IMU Fehler (sofern nicht durch eine redundante IMU Konfiguration kompensierbar) führt zum Versagen aller Zustandsschätzer. Im Normalfall befindet man sich in dem rechten unteren Zustand in Abbildung 4.9, in dem alle drei Filter gleichzeitig arbeiten. Sollte aber z. B. der GNSS Empfang gestört sein[19], wird auf die reduzierten Zustandsschätzer zurückgegriffen. Diese Filter haben im Detail einige Vereinfachungen, erlauben aber einen Betrieb ohne Unterbrechungen, was besonders für Luftfahrzeuge zwingend notwendig ist. Da alle Filter ständig aktiv sind, kann auch ohne Einschwingzeit gewechselt werden. Da das allgemeine 3D-Filter das Accelerometer fortlaufend kalibriert, kann und sollte diese verbesserte Kalibrierung auch an die beiden anderen Filter angebracht werden. Hier ist aber gerade für Fehleranalysen bei Luftfahrtanwendungen darauf zu achten, dass nun

[19] Eine Empfangsstörung kann durch RAIM (Receiver Autonomous Integrity Monitoring) Techniken aufgedeckt werden. Mögliche Verfahren sind z. B.: Konsistenzprüfung der nicht-redundanten Einzellösungen, Pseudorange Residuenanalyse oder statistische Testkriterien nach der Ausgleichung.

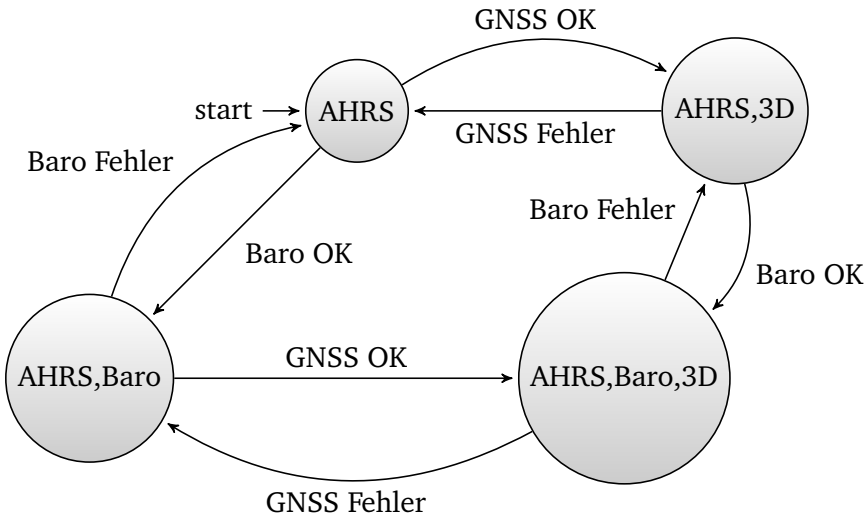


Abbildung 4.9.: Zustandsautomat für die Navigationszustandsschätzung.

die Lagestabilisierung in einer direkten Verbindung mit einem möglichen GNSS Fehler steht: ein GNSS Fehler kann zu einem Biasfehler führen und dieser würde sich dann wieder negativ auf das AHRS auswirken. Der Zustandsautomat erlaubt eine sinnvolle Operation der Navigationszustandsschätzung für verschiedene Anwendungen. Beispielsweise kann innerhalb von Gebäuden, also ohne GNSS Stützung[20], noch die Lage und die Höhe bereitgestellt werden. Dies erlaubt z. B. einem UAV innerhalb von Gebäuden mit einer Lage- u. Höhenregelung zu fliegen.

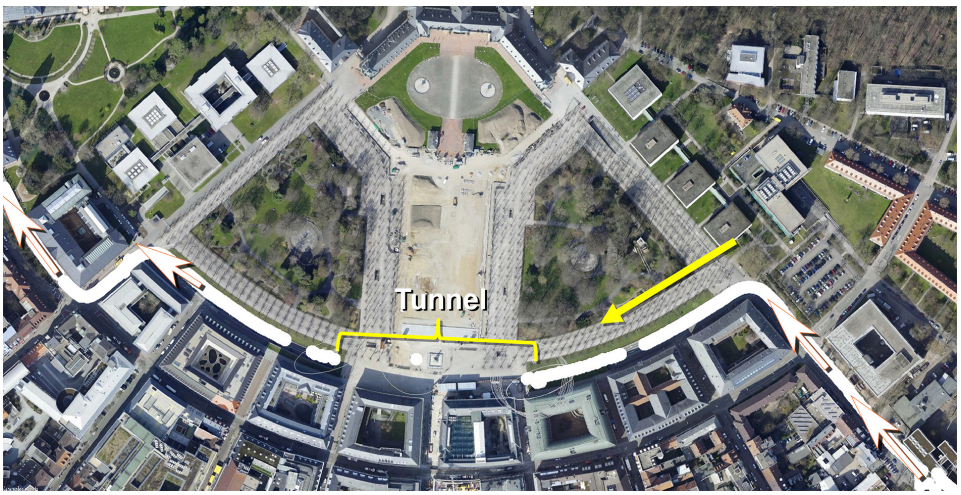
Ein Umschalten muss aber nicht sofort geschehen. Je nach IMU Güte können Ausfälle im 3D Modus über gewisse Zeiträume überbrückt werden (*IMU coasting*). Abbildung 4.10c zeigt zwei Trajektorien, jeweils mit einer eingezeichneten geschätzten Fehlerellipse: von links nach rechts wird der Tunnel verlassen. Die Fehlerellipsen wachsen nach 30 Sekunden ohne GNSS Stützung auf ca. 5 m Radius an, bis am Tunnelende wieder ein GNSS Signal verfügbar ist. Bei der Fahrt von rechts nach links sieht man, wie selbst ohne GNSS Signal im Tunnel anfangs die Inertialnavigation den Ausfall problemlos überbrücken kann. Die Fehler steigen erwartungsgemäß exponentiell an.

Das vorgestellte Navigationskonzept kann selbstverständlich um weitere Sensoren erweitert werden. Allen voran: Kameras. Das gilt für Einzelkameras, Stereokameras und beliebige verteilte Kameras. Die Navigation mit Kameras ist immer noch ein aktuelles Forschungsthema, die Anzahl der Publikationen wächst nach wie vor schnell an. Ein Einstieg findet sich z. B. in Forsyth und Ponce (2011), oder Hirschmüller (2003). Ein guter Überblick über den *State of the Art* bei der Mono-Kamera Navigation wurde von Delmerico und Scaramuzza (2018) veröffentlicht. Wird alleine mit Kameradaten navigiert, so spricht man von Visual Odometry (VO). Werden die Bilder mit IMU Daten fusioniert, so spricht man von Visual Inertial Odometry (VIO). Für die Mono-Kamera Navigation müssen Merkmale (*features*) im Bildsequenzen gefunden werden (siehe Abbildung 4.12). Ist die Abmessung oder die Position der Merkmale nicht bekannt, so muss ein Skalierungsfaktor geschätzt werden, üblicherweise über GNSS oder IMU Beobachtungen (Weiss, 2012). Bekannte Merkmale sind beispielsweise eingemessene Marker bzw.

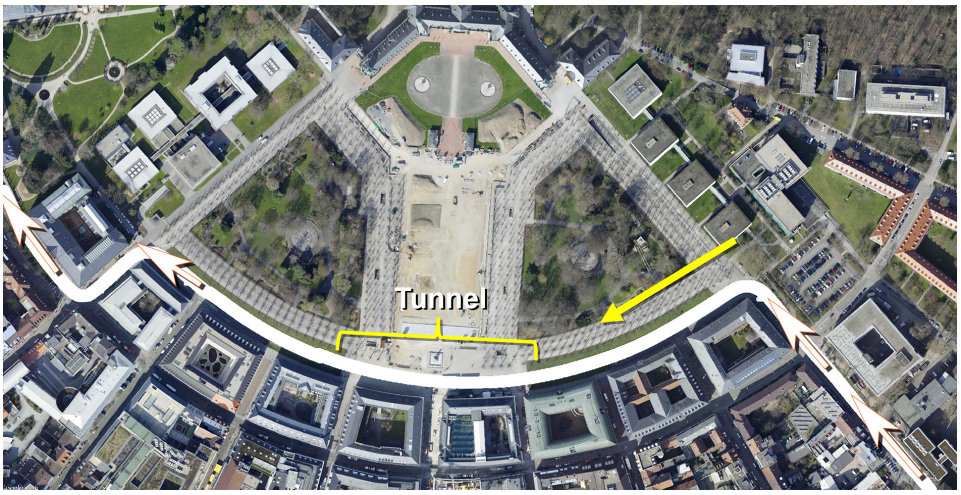
[20] Die GNSS Empfänger sind empfindlich und können auch noch in Gebäuden eine Näherungsposition berechnen, diese eignet sich dann aber nicht mehr für eine Fusion mit IMU Daten; hier müssen für den jeweiligen GNSS Receiver-Typ sinnvolle Grenzwerte ermittelt werden. Besonders die GNSS Geschwindigkeitsgenauigkeit eignet sich dafür gut (Meister, 2010).

Tags. Abbildung 4.11a zeigt die Mono-Kamera Navigation über AprilTags (Olson, 2011). Sowohl beim Feature-Tracking als auch beim Marker-Tracking ist mit häufigen Ausreißern zu rechnen, daher sind robuste Methoden praktisch unumgänglich, Abbildung 4.11b zeigt einen Marker, der an einer Schranktür nicht an der erwarteten Position liegt und daher als Ausreißer verworfen wird. Der Skalierungsfaktor muss bei Stereo-Kamera-Systemen (Abbildung 4.13) nicht geschätzt werden, da dieser durch die kalibrierte Baseline zwischen den Kameras gegeben ist.

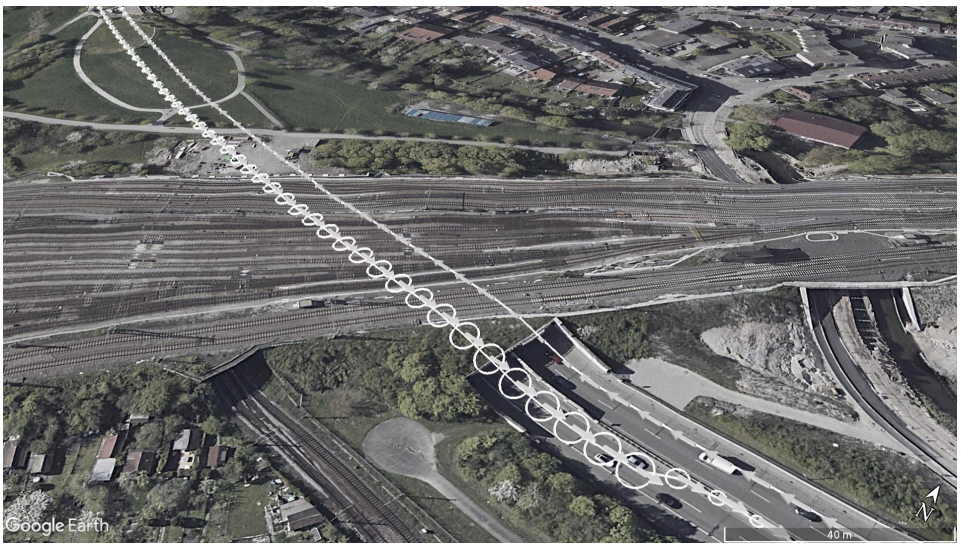
Die Ergebnisse der Forschung in dem Bereich Computer Vision (CV) werden mit Sicherheit in Zukunft weiter Einzug in die Luftfahrt halten, aber die Zulassung von derartigen Systemen ist aufgrund der vielfältigen Fehlermöglichkeiten schwierig.



(a) GNSS Einzelpunktlösung mit häufigen Aussetzern (keine IMU Stützung).

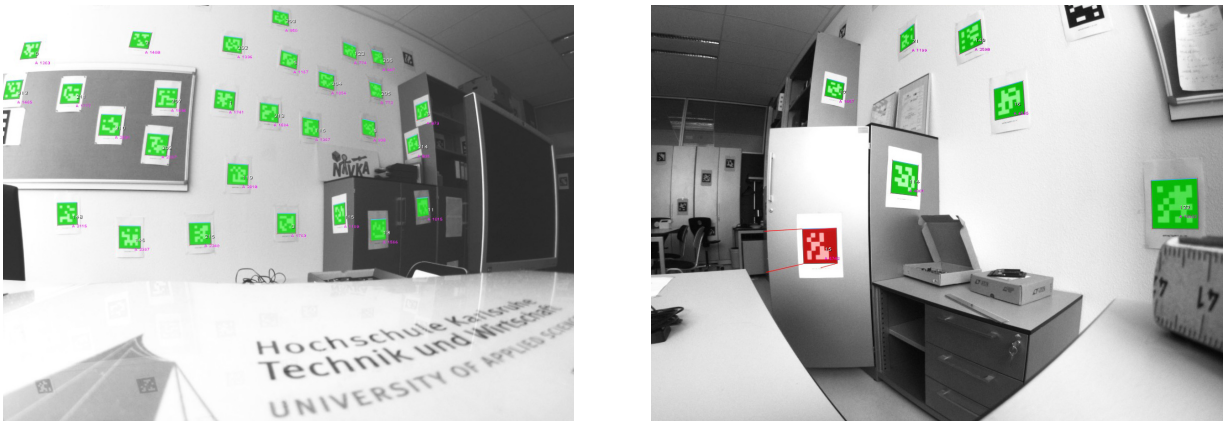


(b) Kontinuierlich verfügbare Navigationslösung mit einem Error-State Kalman-Filter.



(c) Anwachsende Fehlerellipsen von Error-State Kalman-Filter Navigation durch den 609 m langen Karlsruher Edeltrud-Tunnel. Low-Cost IMU InvenSense MPU9150, u-blox M8T GNSS Receiver. Tunneldurchfahrt: 30 Sekunden ohne GNSS Stützung. Barometerstützung aktiv. Fehlerradius bei Tunnelausgang: $1\,\sigma = 5$ m. Datum: 7.8.2015.

Abbildung 4.10.: Typisches Verhalten der GNSS/INS Navigation bei GNSS Signalausfall (IMU *coasting*). Kartenmaterial: GeoBasis-DE/Bundesamt für Kartographie und Geodäsie 2016, Google 2018.



(a) Feld von datumsgebenden (eingemessenen) April-Tags zur hochgenauen (mm-genau) Positionierung.

(b) Outlier Detection über Random Sample Consensus (RANSAC).

Abbildung 4.11.: Mono-Kamera Navigation mit Markern.

Abbildung 4.12.: Mono-Kamera Feature Tracking. Abbildung aus Bloesch et al. (2015).

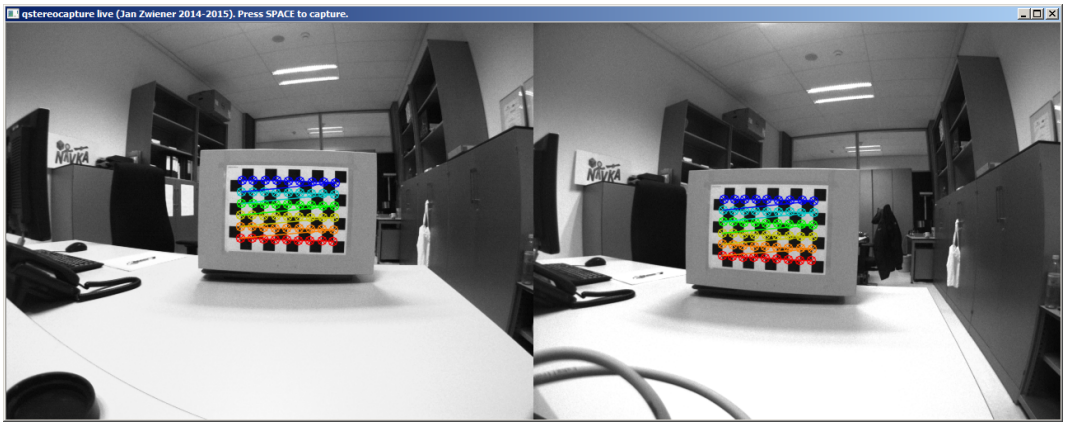


Abbildung 4.13.: Verbund von zwei Kamera zu einem Stereo-Kamera-System (erkannte Punkte der Kalibrierfolie werden farbig markiert).

5 Simplex-erweitertes Kalman-Filter

5.1 Simplex Algorithmus

5.1.1 Einführung

Der Simplex Algorithmus löst lineare Maximierungsprobleme mit Nebenbedingungen. Das Verfahren wurde 1947 von George Dantzig beschrieben. Dieser verwarf es jedoch zuerst wieder, da er das Verfahren fälschlicherweise als zu aufwendig eingeschätzt hatte (Dantzig, 1990). Laut Domschke et al. (2015) stammen die ersten Arbeiten zu dem Thema aber von dem russischen Mathematiker L. V. Kantorovich aus dem Jahr 1939. Der Simplex Algorithmus ist dem Bereich Unternehmensforschung (*engl. Operations Research*) bzw. dem Bereich Lineare Optimierung (*engl. Linear Programming*, kurz LP) zuzuordnen. Also ein wichtiger Zweig der Wirtschaftswissenschaften und der Wirtschaftsinformatik.

Neben den Aufgabenstellungen aus dem Bereich des *Linear Programming* gibt es auch den Bereich der nichtlinearen Optimierung (*Nonlinear Programming*). Zum Beispiel das sogenannte *Quadratic Programming*. Wie in Abschnitt 5.1.6 im Detail gezeigt wird, ist das *Linear Programming* mit der $\mathscr{L}_1$ Norm verknüpft. Dagegen sind die Verfahren aus dem Bereich *Quadratic Programming* mit der $\mathscr{L}_2$ Norm verknüpft. Die wichtigsten Algorithmen für die quadratische Optimierung sind nach Leyffer (2005): das *Active-Set-Verfahren* und das *Interior-Point-Verfahren*.

Der Fokus soll hier aber klar auf dem Bereich der linearen Optimierung liegen, da hier die Robustheit der $\mathscr{L}_1$ Norm genutzt werden soll. Für lineare Optimierungsprobleme gibt es eine Reihe von alternativen Lösungsmethoden, das Simplex Verfahren zählt aber nach wie vor zu den leistungsfähigsten Verfahren (bezüglich des durchschnittlichen Laufzeitverhaltens). Ein guter Überblick dazu findet sich in Domschke et al. (2015). Die zu maximierende lineare Funktion lässt sich in folgender Form darstellen:

$$z = \mathbf{a}^T \cdot \mathbf{y} = a_1 y_1 + a_2 y_2 + \ldots + a_n y_n. \tag{5.1}$$

Der Vektor $\mathbf{y}$ enthält die gesuchten Unbekannten, der Vektor $\mathbf{a}$ beschreibt den linearen Zusammenhang. Es wird der Vektor $\mathbf{y}$ gesucht, der unter gegebenen Bedingungen einen maximalen Wert für z ergibt. Bedingungen werden ebenfalls in linearer Form vorgegeben:

$$c_{i,1} y_1 + c_{i,2} y_2 + \ldots + c_{i,n} y_n \leq b_i. \tag{5.2}$$

Liegen die Ungleichungen in der Form „größer oder gleich“ vor, kann die Ungleichung durch die Multiplikation mit -1 wieder in die Form von Ungleichung 5.2 gebracht werden:

$$c_{i,1}y_1 + c_{i,2}y_2 + \ldots + c_{i,n}y_n \geq b_i \Leftrightarrow -c_{i,1}y_1 - c_{i,2}y_2 - \ldots - c_{i,n}y_n \leq -b_i. \tag{5.3}$$

Der Simplex Algorithmus fügt für jede Ungleichung i eine sogenannte Schlupfvariable (*engl. slack variable*) ein, dadurch wird die Ungleichung in eine Gleichung umgewandelt:

$$c_{i,1}y_1 + c_{i,2}y_2 + \ldots + c_{i,n}y_n + \underbrace{y_{n+i}}_{\text{Schlupfv.}} = b_i. \tag{5.4}$$

Das numerische Ergebnis für y_{n+i} ist unwichtig, solange die Gleichung eingehalten wird. Der Simplex Algorithmus basiert auf dem Aufstellen eines Gleichungssystem, das die gesuchten Variablen und die Schlupfvariablen enthält um dann anschließend mit Hilfe des Gauss-Jordan Verfahrens eine (wenn vorhanden) optimale Lösung zu finden. Das aufgestellte Gleichungssystem wird auch Simplex Tableau genannt.

5.1.2 Simplex Beispielrechnung

Für folgende Zielfunktion

$$z = f(y_1, y_2) = 3y_1 + 4y_2 \tag{5.5}$$

soll unter folgenden Bedingungen das Maximum gefunden werden:

$$2y_1 + 2y_2 \leq 12 \tag{5.6a}$$
$$2y_2 \leq 8 \tag{5.6b}$$
$$0,5y_1 \leq 2 \tag{5.6c}$$
$$y_1, \ldots, y_n \geq 0. \tag{5.6d}$$

Abbildung 5.1 stellt die Bedingungen graphisch dar. Für ein zweidimensionales Problem könnte die Lösung graphisch ermittelt werden, indem die Zielfunktion soweit wie möglich nach rechts oben verschoben wird, ohne den konvexen Lösungsbereich zu verlassen (grau markiert). Siehe dazu auch Domschke et al. (2015) und Abdelmalek und Malek (2008). Die Eckpunkte in Abbildung 5.1 sind mit A, B, C, D, E bezeichnet. Grundlegend ist, dass die gesuchte Lösung an einem der Eckpunkte des konvexen Polyeders (definiert durch die Bedingungen) zu finden ist (Domschke et al., 2015). Vereinfacht gesagt, startet der Simplex Algorithmus an einem Eckpunkt und wandert entlang der Kanten von Eckpunkt zu Eckpunkt, bis eine optimale Lösung gefunden ist.

Je nach Problemstellung kann es auch unmöglich sein, eine Lösung zu finden. Denkbar ist auch, dass die Bedingungen die Lösung nicht genug einschränken, sodass z. B. beliebig große Werte für y_1

und/oder y_2 gewählt werden können, oder strenger formuliert: es handelt sich bei der Lösungsmenge nicht um einen konvexen Polyeder. Ebenfalls ist es möglich, die Ungleichungen so zu formulieren, dass sie sich gegenseitig widersprechen (Abdelmalek und Malek, 2008).

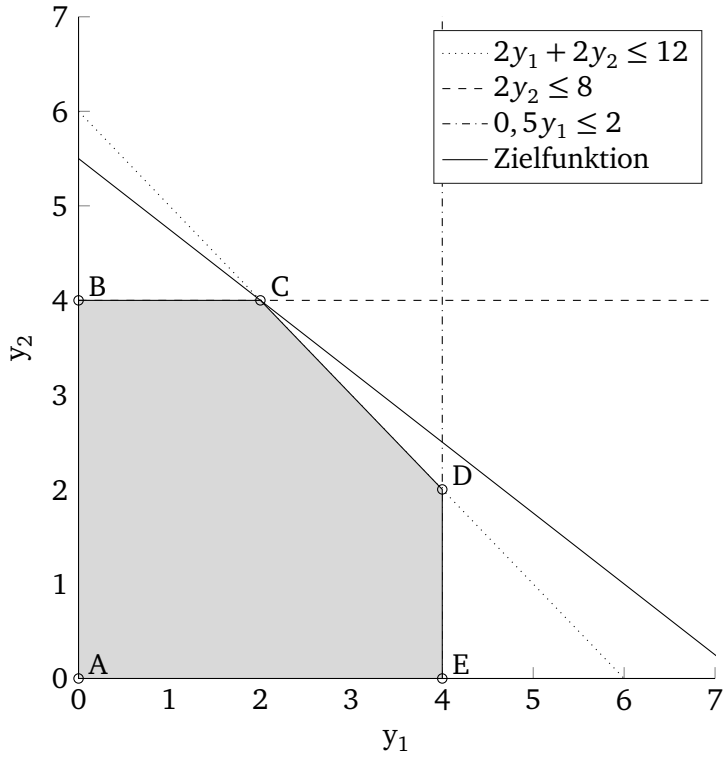


Abbildung 5.1.: Simplex: Visualisierung von Beispielaufgabenstellung.

Das Simplex Tableau sieht folgendermaßen aus:

Basis	y_1	y_2	y_3	y_4	y_5	b
y_3	2	2	1	0	0	12
y_4	0	2	0	1	0	8
y_5	0,5	0	0	0	1	2
z	-3	-4	0	0	0	0

Die letzte Zeile (Ergebniszeile) enthält die zu maximierende Zielfunktion und stellt eine mögliche Startlösung dar (Eckpunkt A). Die Werte (0,0) erfüllen die Bedingungen und sind eine mögliche (wenn auch offensichtlich nicht optimale) Lösung. Abdelmalek und Malek (2008) nennen diese erste Lösung: *basic feasible solution*. Der Wert rechts unten in der Tabelle ist das Resultat der zu maximierenden Zielfunktion. Die Eintragung der Zielfunktionskoeffizienten mit negativem Vorzeichen ist verbreitet (Domschke et al., 2015). Solange negative Elemente in der Ergebniszeile (z) vorhanden sind, wird der Simplex Algorithmus durchlaufen:
Zuerst wird die Pivotspalte gesucht. Das ist die Spalte mit dem kleinsten (negativen) Wert in der Ergebniszeile (z). Also die Unbekannte mit der größten Auswirkung in der Zielfunktion. Das ist vergleichbar mit der Richtung des steilsten Anstiegs *(Gradientenverfahren)*. Das ist im Beispiel die Spalte y_2 mit dem Wert -4.

Basis	y_1	y_2	y_3	y_4	y_5	b
y_3	2	2	1	0	0	12
y_4	0	2	0	1	0	8
y_5	0,5	0	0	0	1	2
z	-3	**-4**	0	0	0	0

Die Werte aus der Spalte b werden jetzt Zeile für Zeile durch den korrespondierenden positiven Wert der Pivotspalte dividiert. Negative Einträge (≤ 0) werden ignoriert. Sind alle Einträge kleiner bzw. gleich 0, so muss das Verfahren an dieser Stelle abgebrochen werden. Es kann keine optimale Lösung gefunden werden (Domschke et al., 2015).

Basis	y_1	y_2	y_3	y_4	y_5	b	
y_3	2	2	1	0	0	12	$12 : 2 = 6$
y_4	0	2	0	1	0	8	$8 : 2 = 4$
y_5	0,5	0	0	0	1	2	$2 : 0 = \lightning$
z	-3	**-4**	0	0	0	0	

Die Zeile mit dem kleinsten Quotienten legt die nächste Pivotzeile fest (hier die Zeile mit dem Quotienten 4). Das Pivotelement ist der Schnitt aus Pivotzeile und Pivotspalte. Alle Werte in der Pivotzeile werden nun durch das Pivotelement dividiert. Damit verlässt eine der Schlupfvariablen (y_4) die Lösungsbasis und wird durch eine der gesuchten Unbekannten ersetzt (y_2). Allgemein gesagt, wird für das Pivotelement die Variable in Zeile und Spalte getauscht. Dieser Schritt hat keine numerische Auswirkung. Er ist aber ein entscheidender Teil des Verfahrens, um am Ende des Verfahrens die Lösung direkt bestimmen zu können.

Basis	y_1	y_2	y_3	y_4	y_5	b
y_3	2	2	1	0	0	12
y_2	0	**1**	0	0,5	0	4
y_5	0,5	0	0	0	1	2
z	-3	-4	0	0	0	0

Nun wird nach dem GAUSS-JORDAN-Algorithmus zu jeder anderen Zeile ein Vielfaches der Pivotzeile addiert, sodass in der Pivotspalte nur noch der Wert 0 vorhanden ist (bis auf das Pivotelement selbst).

Basis	y_1	y_2	y_3	y_4	y_5	b
y_3	2	0	1	-1	0	4
y_2	0	1	0	0,5	0	4
y_5	0,5	0	0	0	1	2
z	-3	0	0	2	0	16

Der Simplex Algorithmus bewegt sich jetzt von einer (i. A. nicht optimalen) Lösung entlang der Kanten zur nächsten (hoffentlich) besseren Lösung. In Abbildung 5.1 hat sich die Lösung nun entlang der Kante von Eckpunkt A zu Eckpunkt B bewegt. Die aktuell beste Lösung ist dadurch von 0 auf 16 angehoben

(Zeile z, Spalte b). Maros (2012) bezeichnet den Simplex Algorithmus auch als *Greedy-Algorithm*, der Schritt für Schritt versucht bessere Ergebnisse zu erzielen.

Nun wird wieder die Spalte mit kleinstem (negativen) Wert in der letzten Zeile gewählt (also den höchsten Kosten). In diesem Beispiel bleibt nur noch y_1 übrig.

Basis	y_1	y_2	y_3	y_4	y_5	b
y_3	2	0	1	-1	0	4
y_2	0	1	0	0,5	0	4
y_5	0,5	0	0	0	1	2
z	-3	0	0	2	0	16

Wieder wird pro Zeile der Quotient aus der Spalte b und der Pivotspalte gebildet. Der kleinste Quotient bestimmt die Pivotzeile.

Basis	y_1	y_2	y_3	y_4	y_5	b	
y_3	2	0	1	-1	0	4	$4 : 2 = 2$
y_2	0	1	0	0,5	0	4	$4 : 0 = \lightning$
y_5	0,5	0	0	0	1	2	$2 : 0{,}5 = 4$
z	-3	0	0	2	0	16	

Die erste Zeile ist somit die neue Pivotzeile und wird durch den Wert des Pivotelements (=2) dividiert.

Basis	y_1	y_2	y_3	y_4	y_5	b
y_1	1	0	0,5	-0,5	0	2
y_2	0	1	0	0,5	0	4
y_5	0,5	0	0	0	1	2
z	-3	0	0	2	0	16

Damit wird auch die Unbekannte y_1 zur Basislösung hinzugezogen und ersetzt die Schlupfvariable y_3. Nach dem Gauss-Jordan Verfahren werden die restlichen Spaltenelemente zu 0 gesetzt.

Basis	y_1	y_2	y_3	y_4	y_5	b
y_1	1	0	0,5	-0,5	0	2
y_2	0	1	0	0,5	0	4
y_5	0	0	-0,25	0,25	1	1
z	0	0	1,5	0,5	0	22

Enthält die Ergebniszeile (z) nur nichtnegative Werte, so ist die aktuelle Basislösung optimal, das Simplexverfahren kann beendet werden (Domschke et al., 2015). In Abbildung 5.1 hat sich die Lösung nun entlang der Kante von Eckpunkt B zu Eckpunkt C bewegt. Die Lösung für y_1 lautet somit 2 (erste Zeile) und für y_2 ergibt sich 4 (zweite Zeile). Das Ergebnis stimmt mit der graphischen Lösung aus Abbildung 5.1 überein. Der (unter den gegebenen Einschränkungen) maximale Funktionswert lässt sich in der Ergebniszeile ablesen: 22.

Allgemein gesagt, kann das Ergebnis für die Variablen in der Basisspalte direkt in der (b) Spalte abgelesen werden. Alle übrigen Variablen haben implizit den Wert 0 angenommen.

Basis	y_1	y_2	y_3	y_4	y_5	b
y_1	1	0	0,5	-0,5	0	2
y_2	0	1	0	0,5	0	4
y_5	0	0	-0,25	0,25	1	1
z	0	0	1,5	0,5	0	22

Der vorgestellte Algorithmus ist auch bekannt als der primale Simplex Algorithmus. Für Problemstellungen, bei denen keine triviale und zulässige Basislösung gegeben ist, kommt der sogenannte duale Simplex Algorithmus zum Einsatz.

5.1.3 Dual Simplex

Der Dual Simplex-Algorithmus muss eingesetzt werden, wenn keine gültige Basislösung vorliegt. Das entspricht dem Fall, dass die Anfangslösung außerhalb des konvexen Lösungspolyeders liegt. Der bisher beschriebene Simplex Algorithmus wird zum sogenannten Dual Simplex erweitert. Jede Iteration im Dual Simplex besteht aus drei Schritten. Diese sind von Domschke et al. (2015) beschrieben als:

- **Schritt 1** (Wahl der Pivotzeile):
  Gibt es kein negatives Element in (b), so muss der duale Algorithmus nicht länger verfolgt werden. Es kann direkt der primale Simplex-Algorithmus durchlaufen werden. Ansonsten wird (statt einer Pivotspalte) eine Pivotzeile gewählt. Die Pivotzeile hat den kleinsten (negativen) Eintrag in der Spalte (b) – die Ergebniszeile ist von dieser Suche ausgenommen.

- **Schritt 2** (Wahl der Pivotspalte):
  Gibt es kein negatives Element in der Pivotzeile, so findet sich keine Lösung. Der Algorithmus muss vollständig beendet werden. Ansonsten wird für jedes Element (ausgenommen sind die Schlupfvariablen) in der Pivotzeile der Quotient mit dem entsprechenden Element aus der Ergebniszeile gebildet. Der höchste Quotient legt das Pivotelement und somit auch die Pivotspalte fest. Ein numerisches Beispiel für diesen Schritt findet sich in Domschke et al. (2014), Abschnitt 2.12 und Domschke et al. (2015), Abschnitt 2.4.2.

- **Schritt 3** (Tableautransformation):
  Das Simplex Verfahren aus Abschnitt 5.1.2 wird auf das Tableau mit dem gewählten Pivotelement angewandt.

5.1.4 Minimierungsprobleme

Für die Anwendung im Rahmen dieser Arbeit ist es notwendig Minimierungsprobleme zu lösen. Die bisher vorgestellte Rechenvorschrift ist auf Maximierungsprobleme ausgelegt (wie sie im Bereich *Operations Research* häufig vorkommen). Zu jedem Optimierungsproblem in der Form

$$\text{Maximiere } z = \mathbf{a}^T \cdot \mathbf{y} \tag{5.7a}$$
$$\mathbf{C} \cdot \mathbf{y} \leq \mathbf{b} \tag{5.7b}$$
$$\mathbf{y} \geq \mathbf{0} \tag{5.7c}$$

gibt es ein entsprechendes Minimierungsproblem:

$$\text{Minimiere } z = \mathbf{b}^T \cdot \mathbf{w} \tag{5.8a}$$
$$\mathbf{C}^T \cdot \mathbf{w} \geq \mathbf{a} \tag{5.8b}$$
$$\mathbf{w} \geq \mathbf{0}. \tag{5.8c}$$

Ein Beweis dazu findet sich z. B. in Domschke et al. (2015). Bei Implementierung des Dual Simplex Verfahrens kann auf entsprechende Umformungen verzichtet werden. Ein Minimierungsproblem kann direkt mit dem Dual Simplex Verfahren gelöst werden (Winston und Goldberg, 2004):

- Bedingungsgleichungen in der Form „größer gleich“ mit Hilfe der Multiplikation mit -1 in die Form „kleiner gleich“ überführt werden.
- Die Zielfunktion mit -1 multipliziert wird.

Dennoch wird die genannte Umformung für viele Anwendungen von Abdelmalek und Malek (2008) empfohlen. Nämlich dann, wenn das ursprüngliche Problem viele Bedingungen und wenige Unbekannte enthält, so ist das umgeformte Problem einfacher zu lösen.

5.1.5 Implementierung und numerische Stabilität

Die vorgestellte Simplexmethode auf Basis des Simplex Tableaus ist nicht immer geeignet für die Lösung mit Hilfe eines Computers (Domschke et al., 2015). Eine Vielzahl von Optimierungsmöglichkeiten wurden publiziert, u. a.:

In Winston und Goldberg (2004) (Kapitel 10) werden verschiedene Ansätze zur Lösung von großen Problemen (*large scale problems*) vorgestellt.

Bartels (1971) beschreibt Stabilitätsprobleme beim Gauss-Jordan Verfahren und schlägt eine Implementierung auf Basis der **LU** Zerlegung vor. Siehe dazu auch Bartels et al. (1971). Der Simplex $\mathscr{L}_1$ Algorithmus von Abdelmalek (1980) basiert auf diesem Ansatz.

In Dantzig und Wolfe (1960) wird beschrieben, wie sich eine Reihe von Problemen in ein Hauptproblem und mehrere Teilprobleme aufspalten lassen um diese getrennt zu lösen. Dabei trägt jede Teillösung

dann zur Verbesserung des Hauptproblems bei. Der große Vorteil ist dabei, dass sich die Teillösungen parallel berechnen lassen. Siehe dazu Tebboth (2001) oder Karbowski (2015).

In Clasen (1966) wird eine Rechenvorschrift zur Vermeidung von Rundungsfehlern beschrieben. Für jedes Ergebnis wird eine spezifische Toleranz berechnet (abhängig von der Fließkommadarstellung). Danach wird geprüft, ob sich das Ergebnis tatsächlich von 0 unterscheidet.

In Storøy (1967) wird eine Methode zum Aufdecken von Fehlern am Ende einer Simplex Berechnung beschrieben.

5.1.6 Lösung von $\mathscr{L}_1$ Problemen mit dem Simplex Verfahren

Barrodale (1968) zeigt numerisch, dass die $\mathscr{L}_1$ Norm gut geeignet ist für Messungen mit groben Fehlern. Späth (1987) vergleicht verschiedene $\mathscr{L}_1$ Schätzalgorithmen hinsichtlich Rechenzeit, Speicherbedarf und Programmcodeumfang (*lines of code*). Der Algorithmus von Barrodale und Roberts (1973) hat hier unter den analysierten Algorithmen den geringsten Speicherbedarf, die geringste Anzahl an Programmzeilen und kann auch mit Rangdefekten in der Designmatrix umgehen. Der Algorithmus von Armstrong et al. (1979) ist dagegen schneller als die übrigen Methoden (Abdelmalek (1980), Bloomfield und Steiger (1980) und Bartels und Conn (1980)).

Es gibt darüber hinaus weitere Ansätze um die $\mathscr{L}_1$ Norm zu berechnen. Beispielsweise über das allgemeine algorithmische Lösungskonzept für M-Schätzer (Jäger et al., 2005), siehe Kapitel 2.5. In Baboolal und Watson (1981) wird für die $\mathscr{L}_1$ Schätzung das iterative Verfahren mit dem Simplex Algorithmus verglichen. Der Simplex Ansatz wird als deutlich schneller beschrieben.

Eine Eigenschaft der $\mathscr{L}_1$ Norm ist, dass sie Punkte aus den Messungen berührt. In Abdelmalek und Malek (2008) wird dies als *Interpolationscharakteristik der $\mathscr{L}_1$ Schätzung* bezeichnet und näher beleuchtet. Der Simplex Algorithmus wird im Detail in Abschnitt 5.1 behandelt.

Unter *Problemen der Lineare Programmierung* versteht man Optimierungsaufgaben, bei denen das Maximum oder Minimum einer linearen Funktion gesucht wird. Das Simplex-Verfahren löst derartige Aufgaben über das GAUSS-JORDAN Eliminationsverfahren (Abdelmalek und Malek, 2008).

In Wagner (1959) wurde gezeigt, dass sich Minimierungsprobleme in der Form

$$\text{minimiere } z = \sum_{i=1}^{n} |v_i| \tag{5.9}$$

$$\mathbf{v} = \mathbf{A} \cdot \mathbf{y} - \mathbf{l} \tag{5.10}$$

so umformulieren lassen, dass sie mit den Techniken der linearen Optimierung bzw. linearen Programmierung gelöst werden können[1]. Das ist auch bekannt als diskrete lineare $\mathscr{L}_1$ Approximation.

In diesem Abschnitt soll nun dargestellt werden, wie die Minimierung der Verbesserungsbetragsfunktion in Gleichung 5.9 entsprechend als lineares Programmierproblem zu formulieren ist. Allgemein kann man sagen, dass die $\mathscr{L}_2$ Norm einfacher zu handhaben ist. Koenker (1997) spekuliert, dass diese schwierige Handhabung der Betragsfunktion auch zum Teil den Erfolg der auf GAUSS zurückgehenden $\mathscr{L}_2$ Norm gegenüber der von LAPLACE gewählten $\mathscr{L}_1$ Norm begründet.

[1] In Wagner (1959) wird zur $\mathscr{L}_1$ Norm auch die Tschebycheff-Norm ($\mathscr{L}_\infty$) behandelt.

Es gibt verschiedene Ansätze das $\mathscr{L}_1$ Problem als LP Problem zu formulieren. Im Folgenden wird die Methode von Barrodale und Roberts (1973) vorgestellt. Alternativen finden sich z. B. in Abdelmalek (1980), Wagner (1959) oder Rabinowitz (1968).

Koenker (1997) gibt für die Zeitkomplexität der Simplex $\mathscr{L}_1$ Schätzung mit n Beobachtungen $\mathcal{O}(n^2)$ an. Diese Aussage bezieht sich auf den $\mathscr{L}_1$ Schätzalgorithmus von Barrodale und Roberts (1974). Koenker (1997) zeigt, dass für kleinere Problemstellungen die $\mathscr{L}_1$ Schätzung schneller berechnet werden kann als z. B. die $\mathscr{L}_2$ Lösung auf Basis der QR Zerlegung. Ab einer bestimmten Problemgröße wird die $\mathscr{L}_1$ jedoch von der $\mathscr{L}_2$ Lösung überholt, da der Zeitbedarf durch die QR Zerlegung nur linear $\mathcal{O}(n)$ mit der Problemgröße wächst. In Farebrother (2013) wird der von Barrodale und Roberts (1974) beschriebene Algorithmus nach wie vor als Mittel der Wahl für $\mathscr{L}_1$ Schätzungen genannt.

Die Kernidee ist es, die Residuen v_i wie folgt auszudrücken:

$$v_i = p_i - q_i \tag{5.11}$$

unter den Bedingungen:

$$|v_i| = p_i + q_i \tag{5.12}$$
$$p_i \geq 0 \tag{5.13}$$
$$q_i \geq 0. \tag{5.14}$$

Beispiel:
Ein Residuenvektor **v** lässt sich unter Einhaltung der genannten Bedingungen darstellen als:

$$\mathbf{v} = \begin{pmatrix} -0,03 \\ -0,16 \\ 0,62 \\ 1,09 \\ 1,10 \end{pmatrix} = \mathbf{p} - \mathbf{q} = \begin{pmatrix} 0,00 \\ 0,00 \\ 0,62 \\ 1,09 \\ 1,10 \end{pmatrix} - \begin{pmatrix} 0,03 \\ 0,16 \\ 0,00 \\ 0,00 \\ 0,00 \end{pmatrix}. \tag{5.15}$$

Für m Beobachtungen und n Unbekannten gilt pro Beobachtung (mit i=1,2,...,m):

$$v_i = \sum_{j=1}^{n} (a_{i,j} \cdot y_j) - l_i = p_i - q_i. \tag{5.16}$$

Da für die Unbekannten y_j im vorgestellten Simplexverfahren implizit die Bedingung $y_j \geq 0$ gilt, kann y_j nicht direkt bestimmt werden. Nach Barrodale und Roberts (1973) lässt sich y_j wieder schreiben als:

$$y_j = b_i - c_j \tag{5.17}$$

mit den Bedingungen:

$$b_j \geq 0 \tag{5.18}$$

$$c_j \geq 0. \tag{5.19}$$

Die zu minimierende Zielfunktion lautet dann:

$$\text{Minimiere} \sum_{i=1}^{m} (p_i + q_i) \tag{5.20}$$

unter den Bedingungen:

$$\sum_{j=1}^{n} \left(a_{j,i} \cdot (b_j - c_j)\right) - (p_i - q_i) = l_i \tag{5.21}$$

$$b_j, c_j, p_i, q_i \geq 0. \tag{5.22}$$

Dieses Problem lässt sich mit dem vorgestellten Dual Simplex Algorithmus lösen. Am Ende hat man $\mathbf{b}, \mathbf{c}, \mathbf{p}$ und $\mathbf{q}$ bestimmt. Aus $\mathbf{b}$ und $\mathbf{c}$ lässt sich der Unbekanntenvektor $\mathbf{y}$ bestimmen. Aus $\mathbf{p}$ und $\mathbf{q}$ die Residuen.

Bisher wurden nur $\geq$ und $\leq$ Bedingungen behandelt. Eine direkte Bedingung wie in 5.21 lässt sich aber über eine $\geq$ und eine $\leq$ Bedingung abbilden. D. h. pro Messung ergeben sich zwei Bedingungen im Simplex Tableau. Beispiel:
Für den Vektor $\mathbf{l}$ soll die $\mathscr{L}_1$ Norm berechnet werden (Median):

$$\mathbf{l} = \begin{pmatrix} 1 & 1 & 1 & 5 \end{pmatrix}^T. \tag{5.23}$$

In Matrixschreibweise:

$$\mathbf{l} = \mathbf{A} \cdot \mathbf{y} \tag{5.24}$$

$$\begin{pmatrix} 1 \\ 1 \\ 1 \\ 5 \end{pmatrix} = \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{pmatrix} \cdot \mathbf{y} = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} \cdot \mathbf{y}. \tag{5.25}$$

Bei diesem einfachen Beispiel, wird somit nur eine Unbekannte y_0 gesucht, bzw. b_0 und c_0. Für die vier Messungen werden 8 positive Residuen gesucht: p_0, q_0, p_1, q_1, p_2, q_2, p_3 und q_3. Die zu minimierende Zielfunktion (siehe Gleichung 5.20) wird durch eine Multiplikation mit -1 in eine zu maximierende Zielfunktion umgewandelt:

$$z = -p_0 - q_0 - p_1 - q_1 - p_2 - q_2 - p_3 - q_3. \tag{5.26}$$

Pro Messungen werden jeweils zwei Bedingungsgleichungen benötigt:

$$a_0 \cdot (b_0 - c_0) - p_0 + q_0 \leq 1 \tag{5.27a}$$
$$a_0 \cdot (b_0 - c_0) - p_0 + q_0 \geq 1 \tag{5.27b}$$
$$a_1 \cdot (b_0 - c_0) - p_1 + q_1 \leq 1 \tag{5.27c}$$
$$a_1 \cdot (b_0 - c_0) - p_1 + q_1 \geq 1 \tag{5.27d}$$
$$a_2 \cdot (b_0 - c_0) - p_2 + q_2 \leq 1 \tag{5.27e}$$
$$a_2 \cdot (b_0 - c_0) - p_2 + q_2 \geq 1 \tag{5.27f}$$
$$a_3 \cdot (b_0 - c_0) - p_3 + q_3 \leq 5 \tag{5.27g}$$
$$a_3 \cdot (b_0 - c_0) - p_3 + q_3 \geq 5 \tag{5.27h}$$

Das dazugehörige Simplex Tableau für den Dual Simplex Algorithmus (Abschnitt 5.1.3) sieht dann folgendermaßen aus:

Basis	b_0	c_0	p_0	q_0	p_1	q_1	p_2	q_2	p_3	q_3	s_0	s_1	s_2	s_3	s_4	s_5	s_6	s_7	b
s_0	1	-1	-1	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1
s_1	-1	1	1	-1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	-1
s_2	1	-1	0	0	-1	1	0	0	0	0	0	0	1	0	0	0	0	0	1
s_3	-1	1	0	0	1	-1	0	0	0	0	0	0	0	1	0	0	0	0	-1
s_4	1	-1	0	0	0	0	-1	1	0	0	0	0	0	0	1	0	0	0	1
s_5	-1	1	0	0	0	0	1	-1	0	0	0	0	0	0	0	1	0	0	-1
s_6	1	-1	0	0	0	0	0	0	-1	1	0	0	0	0	0	0	1	0	5
s_7	-1	1	0	0	0	0	0	0	1	-1	0	0	0	0	0	0	0	1	-5
z	0	0	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0

Die Schlupfvariablen sind in diesem Beispiel mit s_0 bis s_7 gekennzeichnet. Die Schlupfvariablen werden als Ergebnis immer den Wert 0 erhalten. Am Ende errechnet der Dual Simplex Algorithmus korrekt folgendes Ergebnis:

$$y_0 = b_0 - c_0 = 1 - 0 = 1 \tag{5.28a}$$
$$v_0 = p_0 - q_0 = 0 - 0 = 0 \tag{5.28b}$$
$$v_1 = p_1 - q_1 = 0 - 0 = 0 \tag{5.28c}$$
$$v_2 = p_2 - q_2 = 0 - 0 = 0 \tag{5.28d}$$
$$v_3 = p_3 - q_3 = 0 - 4 = -4 \tag{5.28e}$$

Der entscheidende Beitrag von Barrodale und Roberts (1973) ist nun, die Problemstellung durch einen optimierten Algorithmus zu lösen. Dieser kommt mit weniger Iterationen und geringerem Speicherbedarf als der Dual Simplex Algorithmus zu einer optimalen Lösung. Die Anzahl an Simplex Iterationen wird reduziert, indem mehrere benachbarte Simplex Knotenpunkte in einer Iteration durchlaufen werden (Barrodale und Roberts, 1973). Der Quellcode dazu ist in der Programmiersprache FORTRAN veröffentlicht: Barrodale und Roberts (1974). In dem Buch *Numerical Recipes* von Press et al. (2007) wird

aufgrund der Codekomplexität empfohlen, auf eine getestete und veröffentlichte Simplex Implementierung zurückzugreifen. Auch eine Investition in eine kommerzielle Bibliothek kann in manchen Fällen lohnenswert sein (Press et al., 2007, Kap. 10, S. 536).

Zur Aufarbeitung, Validierung und Erstellung der Beispiele der vorgestellten Simplex-Algorithmen wurde im Rahmen dieser Arbeit ein Simplex-Solver mit Web-Oberfläche implementiert, siehe Abbildung 5.2. Für die Auswertung der Ergebnisse in dieser Arbeit wurde dagegen auf das Verfahren von Barrodale und Roberts (1980) aufgesetzt, siehe Abschnitt B.

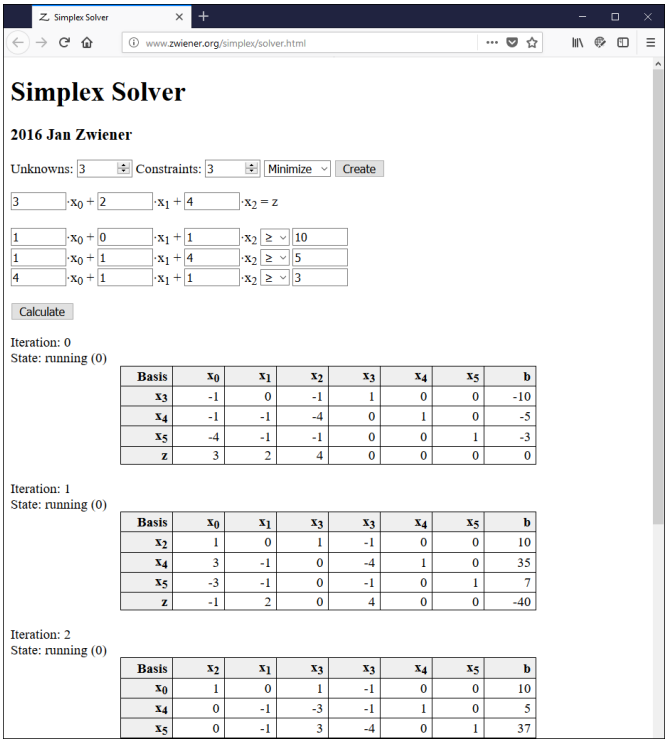


Abbildung 5.2.: Validierung der vorgestellten Simplex Algorithmen über ein JavaScript Programm.

5.1.7 Bestimmung der Varianz-/Kovarianzmatrix

Die $\mathscr{L}_2$ Norm erlaubt die Bestimmung der Kovarianz bzw. Kofaktormatrix direkt aus dem funktionalen Modell, sowie den stochastischen Informationen über die vorliegenden Messungen ($\mathbf{Q}_{ll}$):

$$\mathbf{C}_{yy} = \sigma_0^2 \cdot \mathbf{N}^{-1} \tag{5.29}$$

$$\mathbf{N} = \mathbf{A}^T \cdot \mathbf{Q}_{ll}^{-1} \cdot \mathbf{A}. \tag{5.30}$$

Für die $\mathscr{L}_1$ Norm ist die Bestimmung von $\mathbf{C}_{yy}$ nicht ganz so einfach. Nach Huber (1973)[2] ist die Kovarianzmatrix einer allgemeinen M-Schätzung:

$$\mathbf{C}_{yy} = f^2 \cdot \left(\bar{\mathbf{A}}^T \bar{\mathbf{A}}\right)^{-1}. \tag{5.31}$$

[2] Huber (1973): *„This leads to perfectly horrible and not easily comparable formulas."*

Der Faktor f^2 ist in der Praxis oft nicht zu ermitteln, da zur Berechnung Informationen über den Dichtefunktionsanteil der Störungen benötigt werden (Jäger et al., 2005). Von Jäger et al. (2005) wird die Bestimmung der Kovarianzmatrix aus dem iterativen Algorithmus (siehe Abschnitt 2.5) vorgeschlagen. Im letzten Schritt k der iterativen Methode berechnet sich der gesuchte Vektor $\mathbf{y}_m$ aus dem homogenisierten Modell und der Gewichtsmatrix $\mathbf{W}^k$:

$$\mathbf{y}_m = \mathbf{F} \cdot \bar{\mathbf{l}} \tag{5.32}$$

mit

$$\mathbf{F} = \left(\bar{\mathbf{A}}^T \cdot \mathbf{W}^k \cdot \bar{\mathbf{A}}\right)^{-1} \cdot \bar{\mathbf{A}}^T \cdot \mathbf{W}^k \tag{5.33}$$

und der Gewichtsmatrix $\mathbf{W}^k$, die aus den homogenisierten Residuen gebildet wird

$$\mathbf{W}^k = \mathrm{diag}\left(\left(\frac{1}{|\bar{v}_1|}, \frac{1}{|\bar{v}_2|}, \ldots, \frac{1}{|\bar{v}_n|}\right)^T\right). \tag{5.34}$$

Somit gilt nach einer Fehlerfortpflanzung:

$$\mathbf{C}_{yy} = \mathbf{F} \cdot \mathbf{C}_{\bar{l}\bar{l}} \cdot \mathbf{F}^T = \mathbf{F} \cdot \mathbf{F}^T \tag{5.35}$$

da

$$\mathbf{C}_{\bar{l}\bar{l}} = \mathbf{I}.$$

Dieses Verfahren stellt damit ein wichtiges Fundament dieser Arbeit dar. Nur durch diese stochastische Beschreibung kann der Simplex Algorithmus für die Zustandsschätzung genutzt werden.

Junhuan (2005) schlägt (unter der Annahme von normalverteilten Fehlern) folgende Näherung für die $\mathscr{L}_1$ Norm vor:

$$\mathbf{N}_m = \bar{\mathbf{A}}^T \cdot \bar{\mathbf{A}} \tag{5.36}$$

$$\mathbf{C}_{yy} \approx 2^{-1} \pi \left(\mathbf{N}_m\right)^{-1}. \tag{5.37}$$

Devlin et al. (1981) zeigen, dass für robuste Schätzer die Kovarianzmatrix über eine Monte Carlo Simulation verifiziert werden kann, davon wird in Abschnitt 5.3.3 Gebrauch gemacht.

Bloomfield und Steiger (1980) schreibt, dass die $\mathscr{L}_1$ Schätzung üblicherweise eine (limitierte) Normalverteilung hat. Junhuan (2005) schreibt dazu: „*Unfortunately, the $\mathscr{L}_1$-norm estimate is non-linear, and the basic vector cannot completely be represented as a linear function of the observations, which causes the $\mathscr{L}_1$-norm method to have no complete variance–covariance matrix*“ und zeigt darüber hinaus an einem

Beispiel, wie bestimmte Fehler durch den BIBER-Schätzer (Wicki, 1992) nicht erkannt werden, sehr wohl aber durch die $\mathscr{L}_1$ Lösung.

Es ist an dieser Stelle aber anzumerken, das die von Junhuan (2005) vorgeschlagene Methode in Gl. 5.37 im Rahmen dieser Arbeit in verschiedenen Datensätzen *nicht* zu zufriedenstellenden Ergebnissen geführt hat bzw. zu divergierenden Zustandsschätzungen. Es ist daher fraglich, ob diese Näherungslösung Allgemeingültigkeit besitzt. Dies kann an dieser Stelle jedoch nicht beantwortet werden.

Abbildung 5.3 zeigt zusammenfassend die möglichen Ausprägungen zur Kovarianzbestimmung. Für das Simplex Kalman-Filter (behandelt in Kap. 5.2) wird unterschieden zwischen den verschiedenen Arten, um die Kovarianzmatrix des Zustandsvektors zu berechnen. Die favorisierte und theoretisch begründete Art basiert auf der Gewichtsmatrix $\mathbf{W}^k$. Der Beitrag von Junhuan (2005) konnte, wie beschrieben, nicht in der Praxis genutzt werden.

Eine Möglichkeit sei zusätzlich noch erwähnt. Es hat sich in den praktischen Versuchen gezeigt, dass es für das Simplex Kalman-Filter möglich ist die Formel für das Berechnen der $\mathscr{L}_2$ a posteriori Kovarianzmatrix zu nutzen:

$$\mathbf{Q}_{yy}^{+} = (\mathbf{I} - \mathbf{K} \cdot \mathbf{A}) \cdot \mathbf{Q}_{yy}^{-}. \tag{5.38}$$

Für dieses Vorgehen kann an dieser Stelle keine theoretische Grundlage genannt werden, aber es erscheint dennoch erwähnenswert, dass diese Methode erstaunlich brauchbare Ergebnisse liefert. Rein aufgrund der im Rahmen dieser Arbeit gesammelten Datensätze sowie ohne theoretisches Fundament, kann aber keine allgemeine Empfehlung ausgesprochen werden.

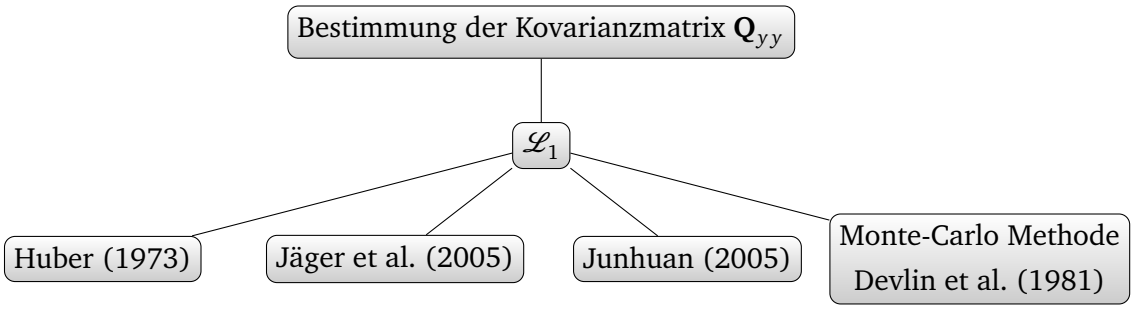


Abbildung 5.3.: Mögliche Ausprägungen $\mathscr{L}_1$ Kovarianzmatrixberechnung.

5.1.8 Ein neuer Algorithmus für die Fehlerfortpflanzung nach einer Simplex Ausgleichung

Auf Basis von Gl. 5.33 u. Gl. 5.34 kann zwar sinnvoll aufgebaut werden; in der praktischen Auswertung hat sich jedoch gezeigt, dass die Form von Gl. 5.33 ungünstig ist. Die Residuen sind von Natur her kleine Werte, oft nahe 0. Eine Division durch 0 ist zwar leicht durch ein Addieren von kleinen Epsilon ϵ Werten zu den Zählern in Gl. 5.34 zu umgehen, aber das Grundproblem bleibt. Tendenziell ist die Matrix $\mathbf{W}^k$ numerisch ungünstig und wird dann in Gl. 5.33 noch mit der Designmatrix $\mathbf{A}$ doppelt multipliziert. Wie in Abschnitt 2.4 betrachtet, ist das Produkt $\mathbf{A}^T \cdot \mathbf{A}$ schon für sich eine numerische Gefahrenquelle. In Kombination mit $\mathbf{W}^k$ hat es sich als echtes Problem bei dieser Arbeit manifestiert. Die Auswertungen in Abschnitt 5.4 wären in dieser Form nicht möglich gewesen.

Als Lösungsvorschlag wird ein Verfahren gezeigt, das dieses Problem umgeht. Im ersten Schritt wird durch Wurzelziehen eine neue Gewichtsmatrix berechnet:

$$\mathbf{W}_w^k = \operatorname{diag}\left(\left(\frac{1}{\sqrt{|\bar{v}_1|}}, \frac{1}{\sqrt{|\bar{v}_2|}}, \ldots, \frac{1}{\sqrt{|\bar{v}_n|}}\right)^T\right). \tag{5.39}$$

Eine Division durch 0 ist nach wie vor als algorithmischer Sonderfall zu umgehen. Durch eine QR-Zerlegung kann nun das Matrixprodukt $\mathbf{A}^T \cdot \mathbf{W}^k \cdot \mathbf{A}$ umgangen werden:

$$\mathbf{A}_w = \mathbf{W}_w^k \cdot \bar{\mathbf{A}} \tag{5.40}$$
$$\mathbf{l}_w = \mathbf{W}_w^k \cdot \bar{\mathbf{l}} \tag{5.41}$$
$$\mathbf{Q}_{l_w l_w} = \mathbf{W}_w^k \cdot \mathbf{W}_w^{k\,T} = \mathbf{W}^k \tag{5.42}$$
$$[\mathbf{Q}_w, \mathbf{R}_w] = \operatorname{qr}(\mathbf{A}_w) \tag{5.43}$$
$$\mathbf{Q}_w \cdot \mathbf{R}_w = \mathbf{A}_w \tag{5.44}$$
$$\mathbf{Q}_w^T \cdot \mathbf{Q}_w = \mathbf{I} \tag{5.45}$$
$$\mathbf{R}_w \text{ ist eine Dreiecksmatrix.}$$

Dies wird nun genutzt um die Kovarianzmatrix $\mathbf{C}_{x_m x_m}$ numerisch stabiler zu berechnen:

$$\mathbf{Q}_w \cdot \mathbf{R}_w = \mathbf{A}_w \tag{5.46}$$
$$\mathbf{Q}_w \cdot \mathbf{R}_w \cdot \mathbf{x} = \mathbf{l}_w \tag{5.47}$$
$$\mathbf{x} = \mathbf{R}_w^{-1} \cdot \mathbf{Q}_w^T \cdot \mathbf{l}_w \tag{5.48}$$
$$\mathbf{x} = \mathbf{R}_w^{-1} \cdot \mathbf{Q}_w^T \cdot \mathbf{W}_w^k \bar{\mathbf{l}} \tag{5.49}$$
$$\mathbf{C}_{x_m x_m} = \mathbf{R}_w^{-1} \cdot \mathbf{Q}_w^T \cdot \mathbf{W}_w^k \cdot \left(\mathbf{W}_w^k\right)^{-T} \cdot \mathbf{Q}_w \cdot \left(\mathbf{R}_w^T\right)^{-1} \tag{5.50}$$
$$\mathbf{C}_{x_m x_m} = \mathbf{R}_w^{-1} \cdot \mathbf{Q}_w^T \cdot \mathbf{W}^k \cdot \mathbf{Q}_w \cdot \left(\mathbf{R}_w^T\right)^{-1}. \tag{5.51}$$

Eine MATLAB Implementierung von Gl. 5.51 ist im Anhang A zu finden.

5.2 Simplex Kalman-Filter

Die grundlegende Idee ist es, ein Kalman-Filter so zu formulieren, dass der gesuchte Zustandsvektor mit Hilfe der Simplex Methode berechnet werden kann. Die Motivation dafür ist:

- Robustheit aufgrund der Simplex $\mathscr{L}_1$ Eigenschaften
- Einfache Einbeziehung von Ungleichungen
- Einfache Einbeziehung von Bedingungsgleichungen

Der Schwerpunkt liegt dabei auf der Navigation, also auf Kalman-Filtern die den Navigationszustandsvektor schätzen. Das vorgestellte Verfahren lässt sich aber allgemein anwenden und ist nicht auf die Navigationszustandsschätzung limitiert.

Der Kunstgriff, um ein Kalman-Filter in Simplex Form zu berechnen, ist die Rückführung in ein allgemeines Ausgleichungsmodell, wie in Gleichung 5.52 beschrieben. Abschnitt 2.8 zeigt, dass die üblichen Kalman-Filter Gleichungen sich aus diesem allgemeinen Modell her ableiten lassen und das Kalman-Filter damit ein Sonderfall der allgemeinen Ausgleichung ist, unter der Annahme, dass zwischen Zustand und Messungen keine Korrelationen bestehen. Die allgemeine Form besteht auf dem einfachen Aufbau: Messungen links, Designmatrix und Unbekanntenvektor rechts und somit mit dem Simplex Algorithmus lösbar, wie in Abschnitt 5.1.6 beschrieben.

$$\underbrace{\begin{pmatrix} \mathbf{y}_k^- \\ \mathbf{l}_k^\star \end{pmatrix}}_{\mathbf{l}} = \underbrace{\begin{pmatrix} \mathbf{I} \\ \mathbf{A}_k^\star \end{pmatrix}}_{\mathbf{A}} \cdot \mathbf{y}_k^+ \tag{5.52}$$

mit der Blockdiagonalmatrix

$$\mathbf{Q}_{ll} = \begin{pmatrix} \mathbf{Q}_{yy,k}^- & 0 \\ 0 & \mathbf{Q}_{ll,k}^\star \end{pmatrix}. \tag{5.53}$$

Nomenklatur für das allgemeine Ausgleichungsmodell:

- $\mathbf{I}$ Einheitsmatrix, die Spalten- und Zeilenzahl entspricht der Anzahl der Zustandsvariablen
- $\mathbf{l}_k^\star$ Messvektor zum Zeitpunkt k
- $\mathbf{Q}_{ll,k}^\star$ Kofaktormatrix für $\mathbf{l}_k^\star$
- $\mathbf{A}_k^\star$ Designmatrix, abgeleitet aus den Beobachtungsgleichungen
- $\mathbf{y}_k^-$ A priori Zustandsvektor
- $\mathbf{y}_k^+$ A posteriori Zustandsvektor
- $\mathbf{Q}_{yy,k}^-$ Kovarianzmatrix der Zustandsschätzung $\mathbf{y}_k^-$

Wie in Abschnitt 2.5 beschrieben, muss das Gleichungssystem noch homogenisiert werden, da sonst die Messungen und Schätzungen nicht entsprechend ihrer geschätzten Genauigkeit gewichtet werden können. Für eine $\mathscr{L}_2$ Ausgleichung wäre dieser Schritt einfach durchzuführen:

$$\mathbf{P} = \mathbf{C}_{ll}^{-1} \tag{5.54}$$

$$\mathbf{N} = \mathbf{A}^T \cdot \mathbf{P} \cdot \mathbf{A} \tag{5.55}$$

$$\mathbf{C}_{xx} = \sigma_0^2 \cdot \mathbf{N}^{-1}. \tag{5.56}$$

Dieser Schritt ist für den Simplex Algorithmus nicht möglich, es wird Abschnitt 2.6 entsprechend ein homogenisierter Messvektor $\bar{\mathbf{l}}$ sowie eine homogenisierte Designmatrix $\bar{\mathbf{A}}$ berechnet:

$$\mathbf{Q}_{ll} = \mathbf{L} \cdot \mathbf{L}^{\mathrm{T}} \tag{5.57}$$

$$\mathbf{P}_{ll} = \mathbf{Q}_{ll}^{-1} \tag{5.58}$$

$$\mathbf{P}_{ll} = \left(\mathbf{L}^{T}\right)^{-1} \cdot \mathbf{L}^{-1} \tag{5.59}$$

$$\mathbf{L}_{w} = \mathbf{L}^{-1}. \tag{5.60}$$

Darauf aufbauend wird homogenisiert:

$$\bar{\mathbf{l}} = \mathbf{L}_{w} \cdot \mathbf{l} \tag{5.61}$$

$$\bar{\mathbf{A}} = \mathbf{L}_{w} \cdot \mathbf{A}. \tag{5.62}$$

Mit der homogenisierten Designmatrix $\bar{\mathbf{A}}$ sowie dem homogenisierten Beobachtungsvektor $\bar{\mathbf{l}}$ kann dann per Simplex Algorithmus ein a posteriori Lösungsvektor $\mathbf{y}_k^+$ berechnet werden (siehe Abschnitt 5.1.6), für die a posteriori Kovarianzmatrix $\mathbf{Q}_{yy,k}^+$ wird auf das Verfahren von Abschnitt 5.1.8 zurückgegriffen. In Anhang B ist der Quellcode für einen optimierten Simplex Solver angehängt, dieser baut auf den Veröffentlichungen von I. Barrodale und F. D. K. Roberts auf (Barrodale, 1968), (Barrodale und Roberts, 1973), (Barrodale und Roberts, 1974) sowie (Barrodale und Roberts, 1980). Damit lässt sich der a posteriori Zustandsvektor berechnen:

$$\mathbf{y}_k^+ = \mathrm{CL1NORM}\left(\bar{\mathbf{A}}, \bar{\mathbf{l}}\right). \tag{5.63}$$

Diese Zusammenhänge führen zu einem neuen Werkzeug, dem Simplex Kalman-Filter. Darauf soll in den folgenden Abschnitten weiter aufgebaut werden. Ungleichungen können direkt in die Schätzung einfließen. Zwangsbedingungen können formuliert werden und darüber hinaus kann ohne Mehraufwand von den robusten $\mathscr{L}_1$ Eigenschaften profitiert werden (vgl. Gl. 5.68). An dieser Stelle soll auch auf die numerisch extrem robuste Implementierung von Barrodale und Roberts (1980) hingewiesen werden: selbst numerisch ungültige Eingaben wie *NAN* oder *INF* im Messvektor werden von CL1NORM problemlos verarbeitet bzw. ignoriert. Jedoch ist darauf zu achten, dass sich bei einer Homogenisierung (beschrieben in Abschnitt 2.6) entsprechende Fehler wie *NAN* und *INF* auf alle Messungen ausstreuen. Wenn der homogenisierte Messvektor nur noch aus ungültigen Zahlen besteht, ist verständlicherweise keine Schätzung mehr möglich.

5.3 Ungleichungen und Bedingungsgleichungen

Einer der Hauptgründe, um das Kalman-Filter in Simplex Form zu modellieren, ist die vergleichsweise einfache Art lineare Ungleichungen zur Einschränkung des Lösungsraumes hinzuzufügen. Natürlich müssen dazu entsprechende Zusammenhänge für die jeweilige Optimierungsaufgabe auch gegeben und verlässlich anwendbar sein. Wenn dies der Fall ist, dann können wahre Fehler minimiert werden und

der Schätzer wird robuster. Zudem können in der Simplex Form auch Bedingungsgleichungen formuliert werden.

Ungleichungen und Bedingungsgleichungen sind aber natürlich kein Alleinstellungsmerkmal des Simplex Verfahrens. Eine Methode um Bedingungsgleichungen in ein $\mathscr{L}_2$ Kalman-Filter einzuführen wird in Simon und Chia (2002) vorgestellt, dabei wird exemplarisch gezeigt, wie Bedingungsgleichungen die Genauigkeit bei der Fahrzeugnavigation steigern können. Simon und Chia (2002) schlägt auch weitere Anwendungsgebiete vor, bei denen die Einschränkung des Zustandsvektors vorteilhaft sein kann und nennt dabei: H_∞ Filterung, Health-Monitoring von Turbofan Triebwerken sowie eben genau die Navigation von Luftfahrtgeräten. In Wang et al. (2002) wird ein Ansatz vorgestellt um harte Bedingungsgleichungen auf Basis der Maximum-Likelihood Methode in ein Kalman-Filter zu integrieren. Dabei wird auch die Möglichkeit *Soft-Constraints* einzufügen diskutiert. *Soft-Constraints* sind virtuelle Beobachtungen, die abhängig von ihren Gewichtungen auch Abweichungen zulassen. Das Konzept der Soft-Constraints kann auch auf die $\mathscr{L}_1$ Schätzung übernommen werden. In Gupta und Hauser (2007) werden verschiedene Algorithmen betrachtet um $\mathscr{L}_2$ Kalman-Filter um Ungleichungen zu erweitern. Es ist jedoch kein triviales Unterfangen, die $\mathscr{L}_2$ Zustandsschätzung dahingehend zu erweitern. Daher werden im Folgenden die Möglichkeit der Simplex Methode betrachtet.

Beispielsweise wären zwei sehr einfache Ungleichungen in einem System, das einen Objektrollwinkel ϕ schätzt:

- $\phi < 10°$
- $\phi > -10°$

Damit wäre sichergestellt, dass sich der geschätzte Winkel zwischen den genannten Grenzen bewegt. Derartiges Vorwissen lässt sich ohne Mehraufwand in einen Zustandsschätzer auf Simplex-Basis integrieren, solange es sich eben als Ungleichung formulieren lässt (siehe 5.1). Die Minimierung der Fehlerbetragssumme (Gleichung 5.64) muss dazu um Ungleichungen erweitert werden.

$$\text{minimiere} \sum_{i=1}^{n} |v_i| \text{ für} \tag{5.64}$$

$$\mathbf{l} + \mathbf{v} = \mathbf{A} \cdot \mathbf{y} \tag{5.65}$$

Wie in Abschnitt 5.1 gezeigt, sind Ungleichungen ein inhärenter Teil des Simplex Verfahrens.

$$\mathbf{E} \cdot \mathbf{y} \leq \mathbf{f}. \tag{5.66}$$

Somit kann der Aufruf der Simplex Kalman-Filter Funktion in Gl. 5.63 erweitert werden zu:

$$\mathbf{y} = \text{CL1NORM}\left(\bar{\mathbf{A}}, \bar{\mathbf{l}}, \mathbf{E}, \mathbf{f}\right). \tag{5.67}$$

An dieser Stelle kann auf die Möglichkeit der Zwangsbedingungen hingewiesen werden. Das Simplex Verfahren erlaubt die direkte Formulierung von Zwangsbedingungen (engl. *linear equality constraints*) in der Form: $\mathbf{C} \cdot \mathbf{y} = \mathbf{d}$. Somit erweitert sich der Funktionsaufruf (optional) zu:

$$\mathbf{y} = \text{CL1NORM}\left(\bar{\mathbf{A}}, \bar{\mathbf{l}}, \mathbf{C}, \mathbf{d}, \mathbf{E}, \mathbf{f}\right). \tag{5.68}$$

Gleichung 5.66 gibt eine für den Zustandsvektor $\mathbf{y}$ einzuhaltende Ungleichung vor. Die Ungleichungen sollten einen nicht-leeren Suchraum aufspannen. Für den Simplex Algorithmus muss zudem eine gültige Initiallösung vorhanden sein (*basic feasible solution*), siehe Abschnitt 5.1.2. Ist der Suchraum so formuliert, dass keine gültige Lösung gefunden werden kann, so wird das im Algorithmus als Abbruchkriterium erkannt (alle Pivotspalteneinträge sind negativ).

Die Einsatzmöglichkeiten für einfache Einschränkungen der Lösung sind vielfältig. Möglich ist z. B. die Bauwerksüberwachung. Hier kann bei einer festen Installation der Sensoren (GNSS und IMU) angenommen werden, dass die Position sich nur in einem kleinen Bereich ändert oder die Geschwindigkeit der Messpunkt sehr klein ist (Gebäudeschwingung). Denkbar wäre auch der Einsatz von Ungleichungen in der Fahrzeugnavigation oder Robotik. Hier können, wie eingangs beschrieben, die Orientierungswinkel (ϕ und θ) für den Betrieb im Nominalzustand eingeschränkt werden.

Eine Sonderstellung für den Bereich Navigationszustandsschätzung hat die Einschränkung der möglichen Position, worauf im nächsten Kapitel eingegangen wird.

5.3.1 Simplex Erweiterung über die Binäre Raumteilung (BSP)

Dieser Abschnitt behandelt die Frage, wie Einschränkungen in der Positionslösung in Ungleichungen formuliert werden können. Für viele Navigationsaufgaben gibt es bekannte Einschränkungen der möglichen Positionen. Beispielsweise könnte für die Fahrzeugnavigation gefordert werden, dass sich die Lösung auf einer Straße befindet und nicht etwa in einem Gebäude. Oder für die Navigationslösung innerhalb eines Gebäudes könnte gefordert sein, dass sich die Position nicht in einer Wand befindet. Die Voraussetzung dafür ist, dass das entsprechendes Kartenmaterial vorhanden ist. Im Folgenden wird angenommen, dass das Kartenmaterial vektorisiert ist und sich eine Topologie daraus ableiten lässt. Damit klar definiert ist, welche Flächen bzw. Räume für den Lösungsraum in Frage kommen und welche ausgeschlossen werden können. Die Verbindung zwischen Kartenmaterial und Ungleichungen wird durch die Geraden bzw. Ebenengleichung nach Hesse[3] hergestellt. Die Kernidee ist, dass durch Ebenengleichungen der Lösungsraum beschrieben und eingeschränkt wird. Abbildung 5.4 zeigt wie eine Gerade bzw. Ebene durch einen Normalenvektor $\mathbf{n}$ und einen Abstand d zum Ursprung dargestellt werden kann. Die Hesse-Normalform eignet sich direkt für die Formulierung in Ungleichungen. Für einen beliebigen Punkt $\mathbf{p_i}$ auf der Ebene gilt:

$$\mathbf{p_i}^T \cdot \mathbf{n} = d. \tag{5.69}$$

[3] Ludwig Otto Hesse (* 1811, † 1874)

Für Punkte auf oder vor der Ebene (in Richtung von $\mathbf{n}$) gilt:

$$\mathbf{p_i}^T \cdot \mathbf{n} \geq d \tag{5.70}$$

sowie

$$-\mathbf{p_i}^T \cdot \mathbf{n} \leq -d. \tag{5.71}$$

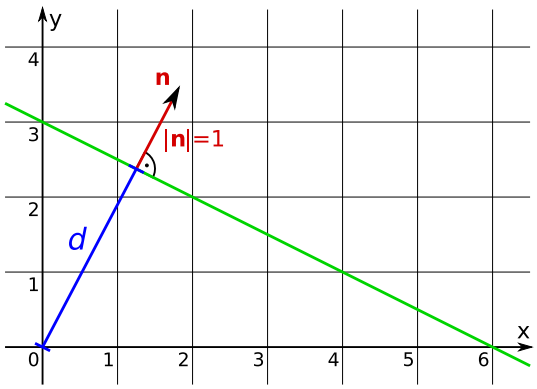


Abbildung 5.4.: Beispielhafte Darstellung einer Ebenengleichung in der Hesse-Normalform.
Bild: *Quartl*, Wikimedia Commons, CC BY-SA 3.0.

Durch die Kombination von verschieden geneigten Ebenen (5.71) in der Designmatrix (5.66) lässt sich ein *Ebenflächner* beschreiben, auch *Polyeder* genannt (Schmidbauer, 2013). Eine Untermenge von $\mathbb{R}^3$, die durch Ebenen begrenzt wird. Polyeder werden somit genutzt um die Positionslösung einzugrenzen. Es wird unterschieden zwischen folgenden Typen:

- Konvexe Polyeder und
- Konkave Polyeder.

Ein einfacher Raum, wie beispielsweise in Abbildung 5.5 zu sehen, lässt sich ideal als konvexer Polyeder in einer Designmatrix beschreiben. Die entsprechenden Ebenengleichungen zu Abbildung 5.5 sind in der dazu passenden Abbildung 5.6 eingezeichnet. Die Normalen ($\mathbf{n}_i$) der Ebenen sind als Pfeile dargestellt und zeigen in die Raummitte. Somit wären für dieses einfache Beispiel nur Positionen innerhalb des Raumes möglich. Die daraus abgeleiteten Ebenengleichungen in Hesse Form sind:

$$\begin{aligned}
d_0 = 1,88,\ \mathbf{n}_0 &= \begin{pmatrix} 0,726 & 0,688 \end{pmatrix}^T \\
d_1 = -1,20,\ \mathbf{n}_1 &= \begin{pmatrix} 0,707 & -0,707 \end{pmatrix}^T \\
d_2 = -4,74,\ \mathbf{n}_2 &= \begin{pmatrix} -0,707 & -0,707 \end{pmatrix}^T \\
d_3 = -1,46,\ \mathbf{n}_3 &= \begin{pmatrix} -0,726 & 0,688 \end{pmatrix}^T
\end{aligned}$$

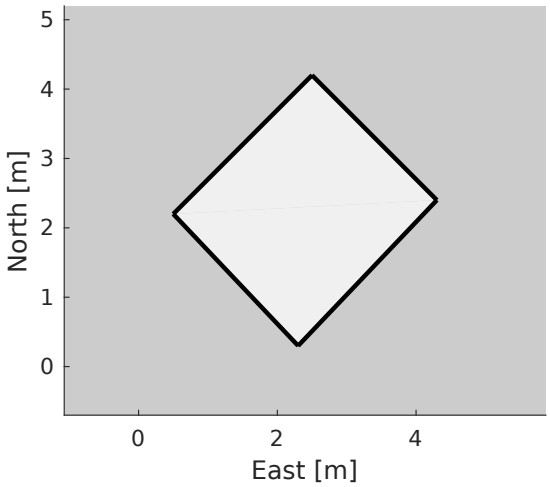


Abbildung 5.5.: Einfacher konvexer Raum (nicht begehbare Fläche in Dunkelgrau, begehbare Fläche in Hellgrau.

Diese Daten können in eine Matrixform übernommen werden, dabei ist nur der Vorzeichenwechsel zu beachten:

$$\mathbf{E} \cdot \mathbf{y} \leq \mathbf{f}$$

$$\underbrace{\begin{pmatrix} -0,726 & -0,688 \\ -0,707 & 0,707 \\ 0,707 & 0,707 \\ 0,726 & -0,688 \end{pmatrix}}_{\mathbf{E}} \cdot \mathbf{y} \leq \underbrace{\begin{pmatrix} -1,88 \\ 1,20 \\ 4,74 \\ 1,46 \end{pmatrix}}_{\mathbf{f}}.$$

Für einen 3D Raum ist eine zusätzliche Spalte für die Z-Achse in Matrix $\mathbf{E}$ notwendig, sowie zwei weitere Zeilen für den Boden und die Decke. Der Vektor $\mathbf{f}$ bleibt ein Spaltenvektor.

Für konkave Räume sind somit keine weiteren Schritte mehr notwendig. In der Praxis muss natürlich für jede Wand die Ebenengleichung automatisiert berechnet werden. Die meisten CAD oder 3D-Design Anwendungen haben aber entsprechende Funktionen integriert bzw. lassen sich durch Plugins oder Skripte erweitern, sodass aus den vorhandenen Raumdaten die Normalen in Hessenormalform extrahiert werden können.

Für eine Verallgemeinerung kann aber nicht nur von konvexen Polyedern ausgegangen werden. Und auch für praktische Anwendungen wäre eine Eingrenzung auf konvexe Polyeder sehr limitierend. Beispielsweise zeigt Abbildung 5.7 einen konkaven Raum. Die Ebenengleichungen in einem konkaven Raum können nicht einfach in einem Schritt bzw. in einer Matrix verarbeitet werden. Die Kombinationen der Ebenengleichungen würden gültige Bereiche ausschließen.

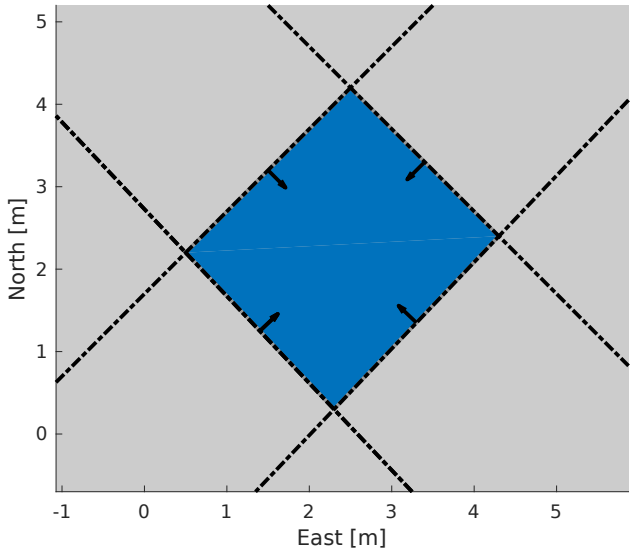


Abbildung 5.6.: Modellierung der Wände über Ebenengleichungen (eingeschlossener begehbarer Bereich in Dunkelblau).

Deshalb soll eine Lösungsstrategie für konkave Räume gezeigt werden. Das Verfahren basiert auf der *binären Raumteilung*, engl. Binary Space Partitioning (BSP). Die Grundidee ist, dass der konkave Raum solange aufgeteilt wird, bis nur noch konvexe Subräume übrig sind (siehe *divide and conquer* Strategie).

In Schumacker et al. (1969) wird die binäre Raumteilung beschrieben um die Berechnung von Grafiken für einen Flugsimulator zu beschleunigen, geht aber von einer manuellen Vorbearbeitung bzw. Raumteilung aus. Eine automatisierte binäre Raumteilung wird in Fuchs et al. (1980) beschrieben, aber ebenfalls mit einem klaren Fokus auf den Bereich der Computergrafik. In der Dissertation von Teller (1992) wird das Verfahren im Detail beschrieben und genutzt um eine i. A. konkave Raumgeometrie in konvexe Subräume zu teilen und die Sichtbarkeit als statische Lookup-Tabelle vorausberechnen. Statt einer kompletten konkaven Raumgeometrie werden hier jeweils nur die sichtbaren konvexen Subräume gezeichnet.

Die binäre Raumteilung kann nun für die vorliegende Problemstellung wie folgt genutzt werden:

- Gegeben sei ein konkaver Polyeder, der gültige Positionen für eine Navigationszustandsschätzung beschreibt.

- Dieser konkave Polyeder wird entlang einer Wand geteilt. Bei dieser Teilung entstehen zwei Subräume. Jeder dieser Subräume wird nun wieder auf die gleiche Art geteilt (Rekursion). Ist ein resultierender Subraum konvex, wird dieser nicht weiter geteilt. Am Ende liegen nur noch konvexe Subräume vor. Die Aufteilung erzeugt eine Baumstruktur und kann statisch abgelegt werden. Durch den Schnitt durch die Geometrie müssen bestehende Wände geteilt werden und in den jeweiligen Subraum zugeordnet werden, dadurch erhöht sich die Komplexität der Geometrie.

- Im Zuge der Teilung erfolgt gleichzeitig die Berechnung der Ebenen (in Hessescher Normalenform), die für jeden konvexen Subraum zugehörig sind.

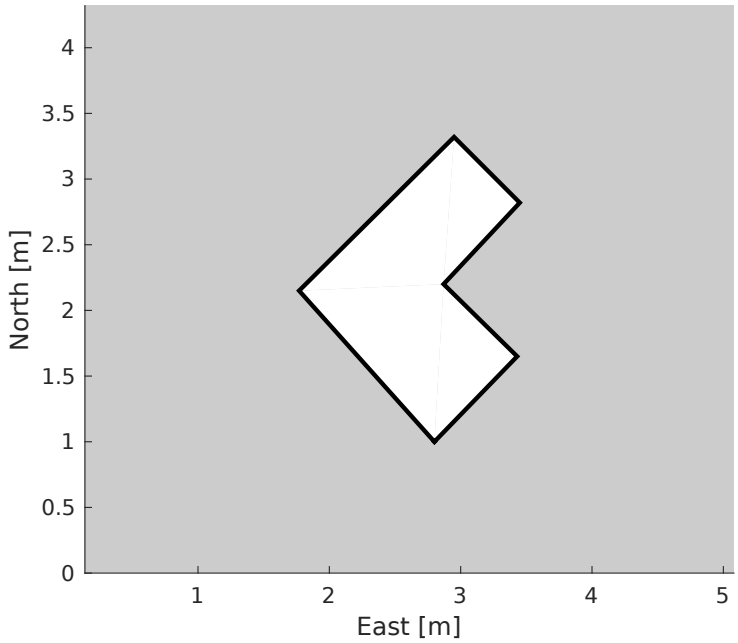


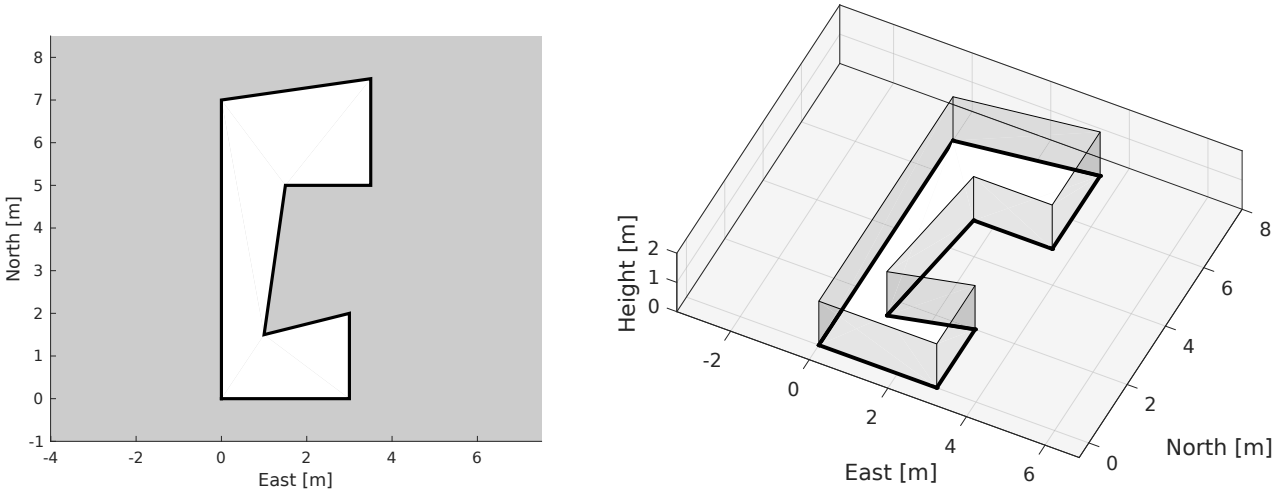
Abbildung 5.7.: Konkaver Raum.

- Nun kann eine Simplex $\mathscr{L}_1$ Lösung für jeden konvexen Subraum berechnet werden. Die Lösung mit der niedrigsten Fehlerbetragssumme ist zu wählen.

Hervorzuheben ist die Wahl der Polyederwand für die Aufteilung. Die gewählte Trennwand soll zwei Subräume mit möglichst gleicher Anzahl an Wänden vor der Teilungsebene (W_f) und hinter der Teilungsebene (W_b) erzeugen (*balanced tree*), sowie möglichst wenig Schnitte (S) durch weitere Wände erzeugen. Die dazugehörige Kostenfunktion C für die mögliche Trennwand i lautet:

$$C(i) = |W_f - W_b| + S. \tag{5.72}$$

C muss in jedem Rekursionsschritt für alle möglichen Trennwände i berechnet werden. Die Wand mit der besten Bewertung (kleinster Wert für C) wird dann als Trennwand ausgewählt. Teller (1992) beschreibt diese Methode in Abschnitt 5.2.1 *Splitting Criteria*. Während die Offline-Generierung mit Aufwand verbunden ist, so kann diese Datenstruktur sehr gut mit wachsender Komplexität der Geometrie umgehen. Ein BSP Baum ist mit einer sortierten Liste vergleichbar und hat für das Durchsuchen eine $\mathscr{O}(\log n)$ Charakteristik.



(a) 2D Ansicht **(b)** 3D Ansicht

Abbildung 5.8.: Konkaver Beispielraum (nicht begehbare Fläche in Dunkelgrau, Wände in Schwarz und begehbare Fläche in Weiß).

5.3.2 Beispiel für Binäre Raumteilung (BSP)

Das BSP Simplex Verfahren soll anhand eines konkaven Polyeders in der Ebene verdeutlicht werden. Abbildung 5.8 zeigt die Geometrie für das folgende Beispiel. Dunkelgraue Flächen sind nicht begehbar und von der Lösungsmenge auszuschließen.

In **Schritt 1** (Abbildung 5.9) wird die erste Ebene zur Teilung in zwei Subräume gewählt. Die Auswahl erfolgt über die Kostenfunktion 5.72. Die Trennebene ist mit einer gestrichelten Linie sowie einem Normalenvektor markiert. Der Bereich vor der Trennebene liegt in der Richtung des Normalenvektors. Die Anzahl der Wände vor der Schnittebene ist: $W_f = 4$, hinter der Schnittebene $W_b = 5$. Die Anzahl der geteilten Wände S ist 1, da die linke Wand durch die Trennebene geteilt wird. Somit beträgt die Kostenfunktion C für die Trennwand $|W_f - W_b| + S = 2$. Aufgrund der Symmetrie in diesem Beispiel gäbe es auch alternative Trennebenen mit den gleichen Kosten C. Bei gleicher Bewertung kann eine beliebige Wand gewählt werden, da das Ergebnis letztlich gleichbleibt. Alle weiteren möglichen Trennebenen haben einen höheren Wert C der Kostenfunktion.

Schritt 2 (Abbildung 5.10): Nach der Teilung ist der obere Subraum (magentafarbene Wände) bereits konvex und muss nicht weiter geteilt werden. Um zu prüfen, ob ein Subraum konvex ist, wird für jede Wand berechnet, ob alle übrigen Wände vor dieser Wand liegen. Im 2D-Fall ist dies besonders einfach. Jede Wand lässt sich durch zwei Eckpunkte beschreiben. Dann wird der Abstand dieser Eckpunkte von der jeweiligen Ebene berechnet. Also ein Vektorgeometrieproblem (*Point-Plane Distance*), das z. B. in Lengyel (2004) beschrieben wird. Sind alle Abstände in allen Varianten positiv, dann ist der Subraum konvex.

Somit muss der Algorithmus nur noch auf den unteren Subraum angewandt werden. Hier entsteht wieder ein linker (rote Wände) und ein rechter Subraum (blaue Wände).

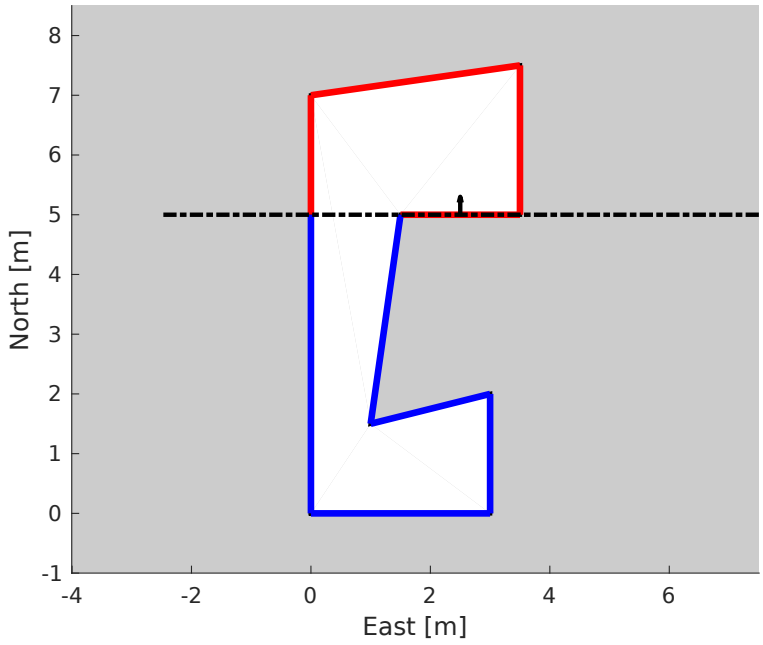


Abbildung 5.9.: Binäre Raumteilung für konkaven Beispielraum (Schritt 1).

Schritt 3 (Abbildung 5.11): Nach der letzten Teilung gibt es nur noch konvexe Subräume. Somit kann der letzte Zweig des rekursiven Algorithmus beendet werden. Es sind drei konvexe Subräume entstanden (Subraum A, Subraum B und Subraum C).

Diese Teilung lässt sich als Binärbaum darstellen (Abbildung 5.13) mit zwei Schnittebenen (Abbildung 5.12). Der linke Ast verzweigt auf die Wände, die vor der Schnittebene liegen ($\mathbf{p} \cdot \mathbf{n} \geq d$). Der rechte Ast zeigt auf die Wände hinter der Schnittebene ($\mathbf{p} \cdot \mathbf{n} < d$).

Diese Baumdarstellung erweist sich als nützlich. Das Ziel der Aufteilung in konvexe Subräume ist es, dass die Ebenen, die den jeweiligen Subraum beschreiben, in die Designmatrix für die Ungleichungen übernommen werden. Aus der Baumstruktur können für die Subräume die benötigten Ebenengleichungen direkt entnommen werden.

Ein wichtiger Punkt ist hier, dass nicht einfach nur die von den Wänden abgeleiteten Ebenen benötigt werden, sondern auch die Teilungsebenen. Sonst wären die Subräume nicht geschlossen und damit auch keine Polyeder mehr. Die Ungleichungen in der Designmatrix würden den Suchraum dann nur ungenügend einschränken.

Der Binärbaum enthält alle Ebenen und muss für jeden Subraum traversiert werden um die relevanten Ebenen zu entnehmen. Eine allgemeine Binärbaumtraversierungsmethode, spielt keine Rolle (*pre-order, post-order, in-order*, etc.). Es wird einfach von einem Subraum, also einem Binärbaumblatt, ausgegangen und linear bis zum Wurzelknoten aufgestiegen. Dabei wird die Ebene von jedem besuchten Knoten in die Designmatrix aufgenommen. Verzweigungen, die hinter einer Schnittebene liegen übernehmen diese Ebene mit invertierter Richtung, damit der Subraum geschlossen bleibt. Ansonsten würde die Ebenengleichung in der Designmatrix in die falsche Richtung zeigen.

Für den Binärbaum aus Abbildung 5.13 ergeben sich folgende relevante Ebenen:

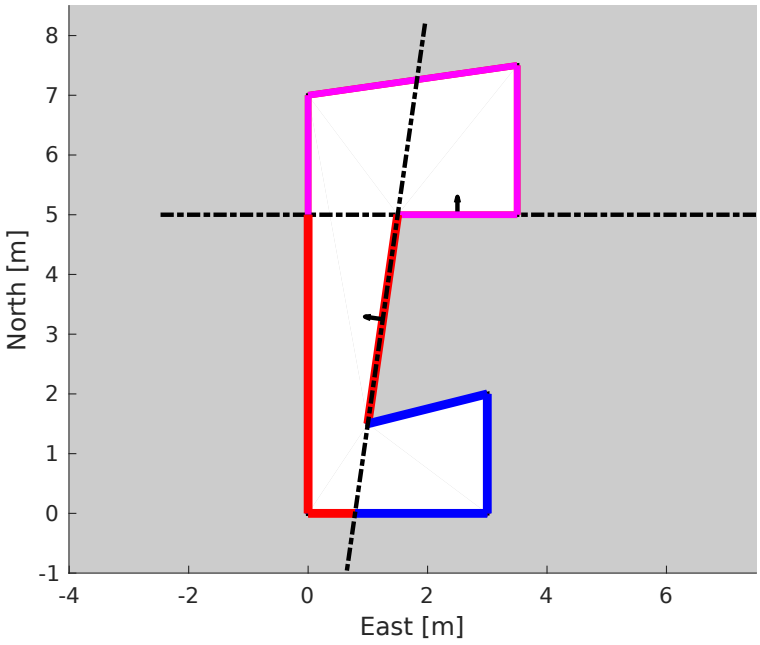


Abbildung 5.10.: Binäre Raumteilung für konkaven Beispielraum (Schritt 2).

- Menge der Ebenen für konvexen Subraum A: {**A**, **1**}
- Menge der Ebenen für konvexen Subraum B: {**B**, **2**, -**1**}
- Menge der Ebenen für konvexen Subraum C: {**C**, -**2**, -**1**}

Das negative Vorzeichen vor der Ebenennummer markiert die invertierte Richtung der Ebene. Zur Verdeutlichung sind die Ebenen der Subräume in Abbildung 5.14a, 5.14b und 5.14c dargestellt.

Im nächsten Schritt kann nun das eigentliche Ausgleichungsproblem berechnet werden. Dazu werden 30 normalverteilte Positionsmessungen an einer Stelle eingefügt, die gleich die relevanten Sonderfälle hervorhebt. In Abbildung 5.15 sind die $\mathscr{L}_1$ Lösungen für die Subräume A und B in roter Farbe dargestellt. Die $\mathscr{L}_1$ Lösung für den Subraum C hat die niedrigste Fehlerbetragssumme von allen Subräumen und ist in blau markiert. Die Positionsmessungen sind als schwarze Punkte zu sehen. Die relevanten Sonderfälle sind:

- Die Messungen verteilen sich auf verschiedene Subräume an einer konkaven Stelle.
- Die Messungen verteilen sich zum Teil auch auf ungültige Bereiche (grau markiert).
- Der Schwerpunkt der Messungen liegt im ungültigen Bereich.

In der nachfolgenden Tabelle sind die Fehlerbetragssummen der einzelnen $\mathscr{L}_1$ Lösungen aufgelistet. Jede Zeile ist das Ergebnis einer Simplex $\mathscr{L}_1$ Ausgleichung mit den 30 normalverteilten Positionsmessungen sowie den jeweiligen Ungleichungen aus den Ebenengleichungen. Der gesuchte Zustandsvektor $\mathbf{y}$ besteht nur aus einer Rechts- und Hochwertkomponente. Somit besteht die Designmatrix $\mathbf{A}$ nur aus einer Einheitsmatrix.

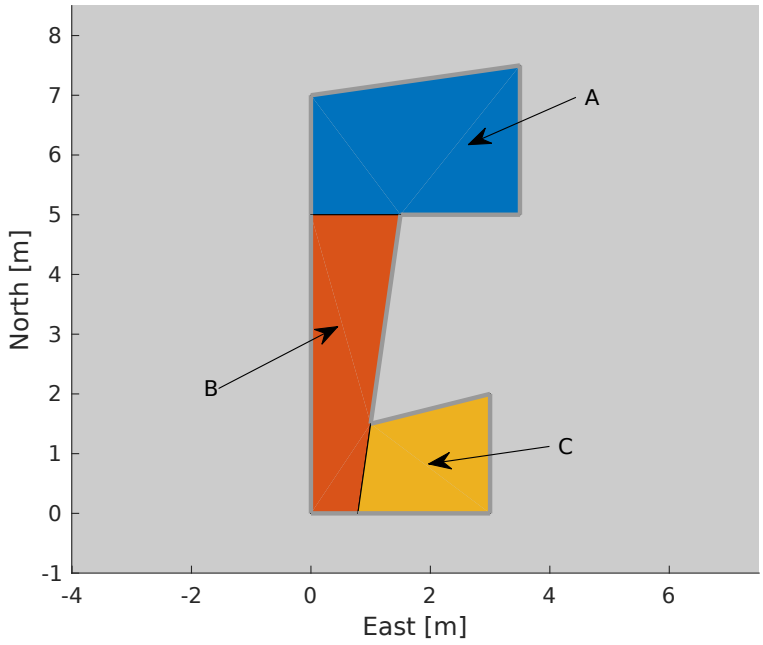


Abbildung 5.11.: Binäre Raumteilung für konkaven Beispielraum (Schritt 3).

Subraum	$\sum_{i=1}^{n} \lvert v_i \rvert$ (m)
A	1126.41
B	102.95
C	69.54

Subraum C hat die niedrigste Fehlerbetragssumme, somit ist der Lösungsvektor aus dieser Ausgleichung zu wählen. Subraum A hat eine vergleichsweise hohe Fehlerbetragssumme, was aber auch in Abbildung 5.15 deutlich zu sehen ist. Denkbar wäre, dass der Algorithmus an dieser Stelle um eine Heuristik erweitert wird, sodass Subräume, die weit von den Messungen entfernt liegen, gar nicht erst berücksichtigt werden. Die Lösung von Subraum B liegt direkt auf einer Ebene, da der Schwerpunkt der Messungen hinter dieser Ebene liegt.

Randbemerkung:

Das Verschieben der Ebenen (über den d Parameter in der Hessenormalform) erlaubt es die Lösungen in einer bestimmten Wandnähe auszuschließen. Z. B. für Anwendungsfälle in denen die Objektausdehnung bekannt ist und sich als Kugel mit einem bestimmten Radius r modellieren lässt. Dann können die Ebenen um diesen Radius r in Richtung Raummitte geschoben werden. Abbildung 5.16 zeigt die Verschiebung der Ebenen in dem blauen Subraum. Vorsicht ist an den *Portalübergängen* zu den anliegenden Räumen geboten, da hier Streifen entstehen, die aus dem Lösungsraum ausgeschlossen werden. Falls das für den jeweiligen Anwendungsfall akzeptabel ist, ergibt sich damit eine einfache Möglichkeit, die Objektausdehnung direkt in die Zustandsschätzung selbst miteinzubeziehen. Ansonsten muss der Übergangsstreifen als eigener konvexer Raum erzeugt werden, was im Rahmen dieser Arbeit nicht verfolgt wurde.

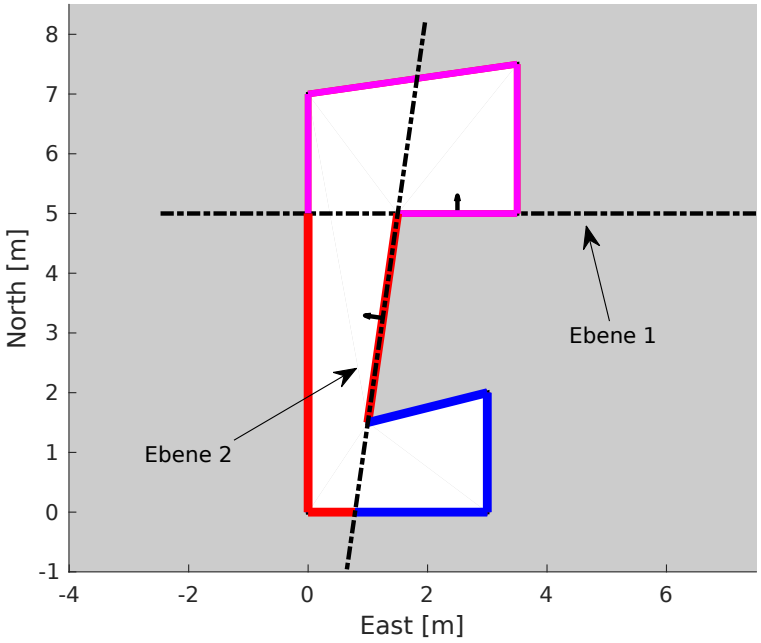


Abbildung 5.12.: Schnittebenen für konkaven Beispielraum.

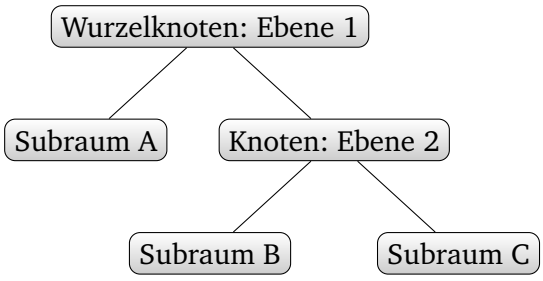
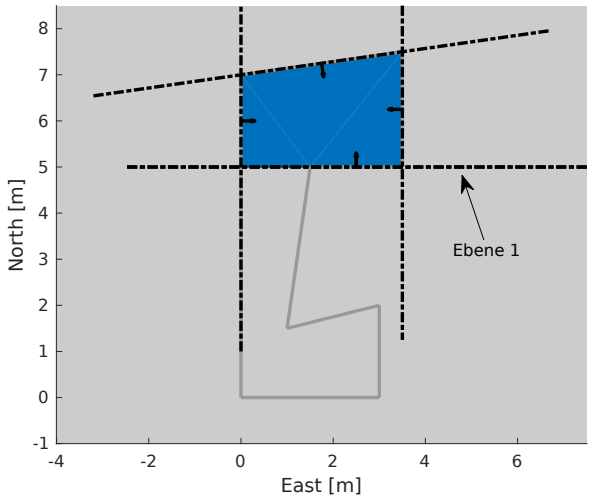
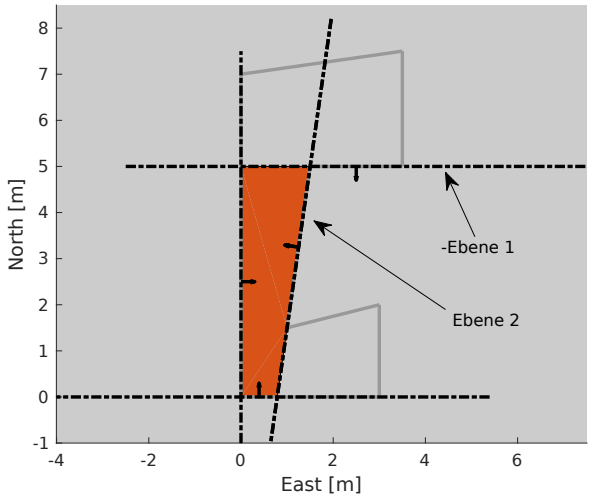


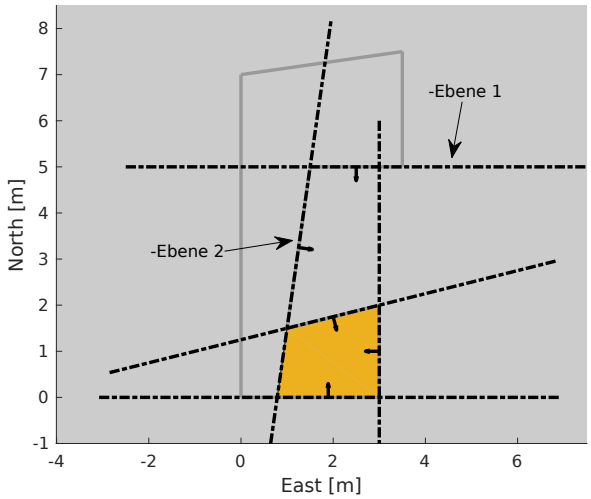
Abbildung 5.13.: Binärbaumdarstellung für konkaven Beispielraum.



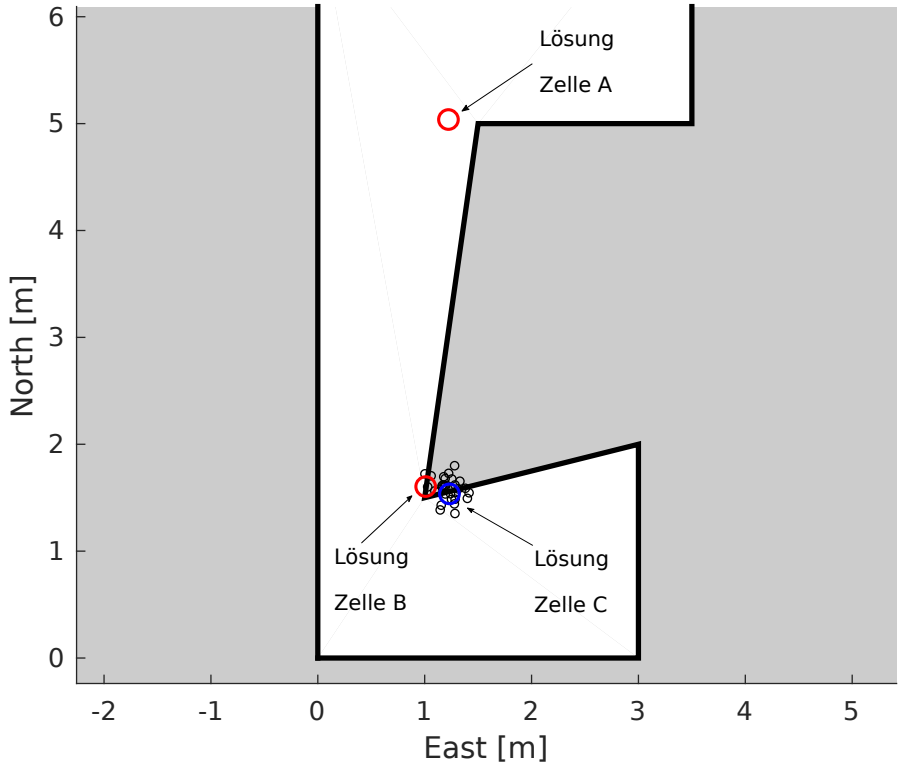
(a) Ebenen für Subraum A, abgeleitet aus der binären Baumstruktur.



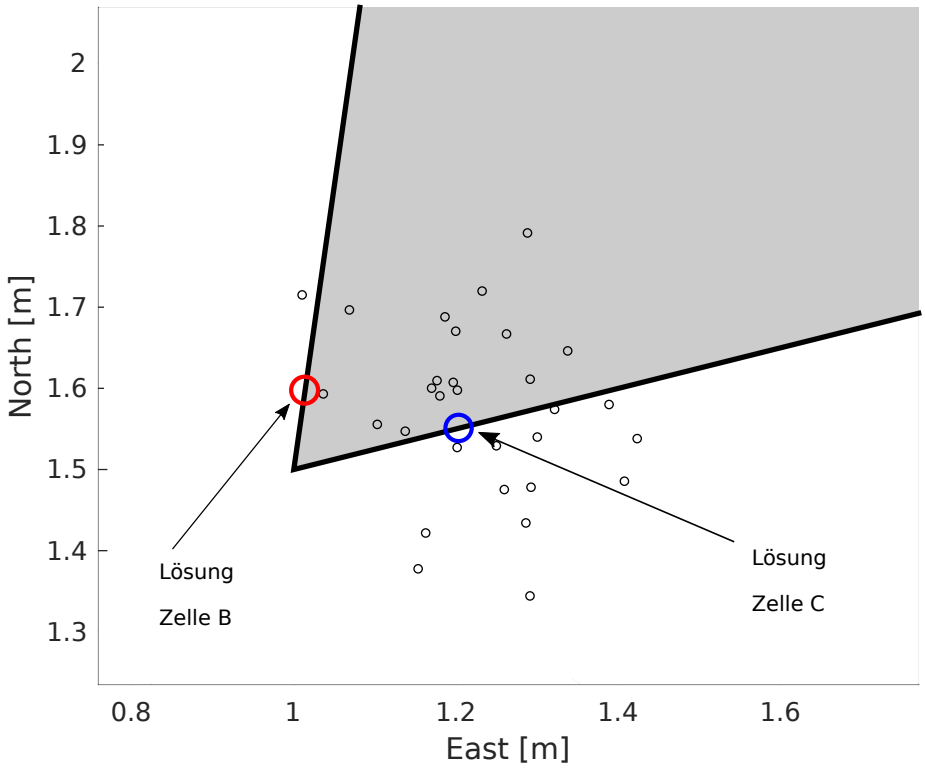
(b) Ebenen für Subraum B, abgeleitet aus der binären Baumstruktur (Ebene 1 mit invertierter Richtung).



(c) Ebenen für Subraum C, abgeleitet aus der binären Baumstruktur (Ebene 1 und Ebene 2 mit invertierter Richtung; Ebene 1 wäre in diesem Fall nicht notwendig, im Allgemeinen aber schon).



(a) Gesamtüberblick.



(b) Zoom auf die normalverteilten Messungen.

Abbildung 5.15.: In roter Farbe: $\mathscr{L}_1$ Lösung die Zellen A u. B. Die $\mathscr{L}_1$ Lösung für Zelle C in blau (niedrigste Fehlerbetragssumme). Schwarze Punkte: Messungen.

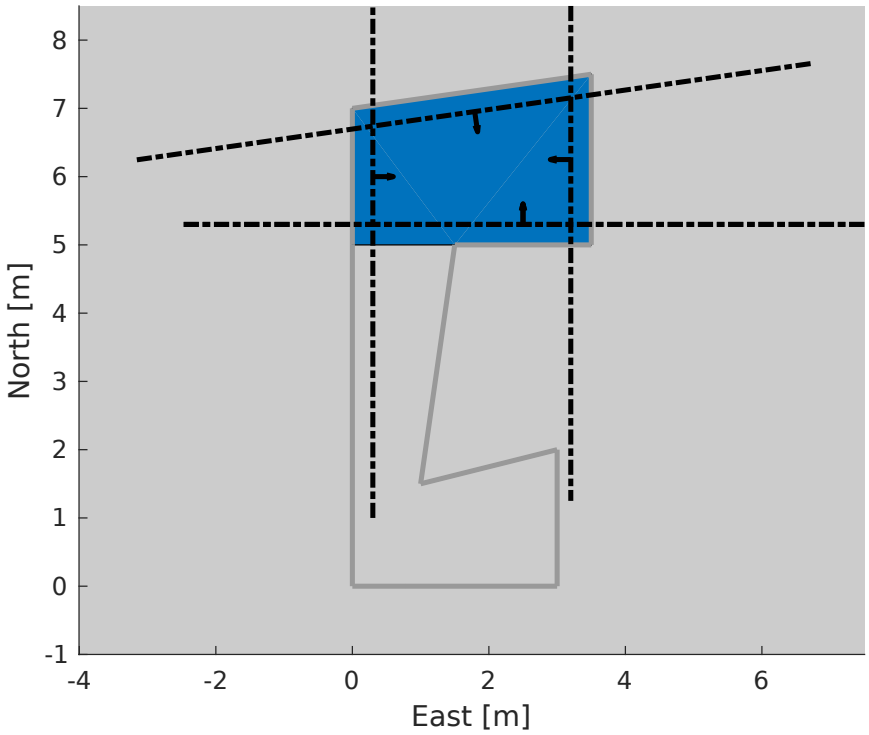


Abbildung 5.16.: Verschiebung der Ebenennormalen zur Einschränkung des Lösungsraumes.

5.3.3 Einfluss der Ungleichungen auf die Effizienz des Schätzers

Es ist möglich, dass durch harte Restriktionen hohe Residuen auftreten. Beispielsweise durch Wände, welche die Positionsmessungen mit großem Abstand von dem möglichen Lösungsraum trennen.

Somit berechnet sich die Kovarianzmatrix $\mathbf{F} \cdot \mathbf{F}^T$ anhand von der Gewichtsmatrix aus dem letzten Iterationsschritt i:

$$\mathbf{F} = \left(\bar{\mathbf{A}}^T \cdot \mathbf{W}^i \cdot \bar{\mathbf{A}}\right)^{-1} \bar{\mathbf{A}}^T \cdot \mathbf{W}^i \tag{5.73}$$

mit hohen homogenisierten Residuen $\bar{\mathbf{v}}$ in der Gewichtsmatrix $\mathbf{W}$:

$$\mathbf{W}^i = f(\bar{\mathbf{v}}), \tag{5.74}$$

also für den Fall der $\mathscr{L}_1$ Norm mit:

$$\mathbf{W}^k = \mathrm{diag}\left(\left(\frac{1}{|\bar{v}_1|}, \frac{1}{|\bar{v}_2|}, \ldots, \frac{1}{|\bar{v}_n|}\right)^T\right). \tag{5.75}$$

Es wäre ein Trugschluss, von großen Residuen auf eine schlechte Kovarianzmatrix zu schließen. Bei Gl. 5.73 würden sich beispielsweise bei einer Einheitsdesignmatrix $\bar{\mathbf{A}} = \mathbf{I}$ die Fehler aufheben. Werden in der Nähe einer Wand normalverteilte Messungen (vergleichbar mit Abbildung 5.15) verarbeitet, ergibt sich folgende Beispielkovarianzmatrix:

$$\mathbf{Q}_{yy,BSP} = 10^{-3} \cdot \begin{pmatrix} 0.13 & 0.02 \\ 0.02 & 0.03 \end{pmatrix}, \tag{5.76}$$

wird dagegen eine Lösung ohne Einbeziehung von Ungleichungen berechnet, so ergibt sich folgende Kovarianzmatrix:

$$\mathbf{Q}_{yy} = 10^{-3} \cdot \begin{pmatrix} 0.13 & 0.02 \\ 0.02 & 0.25 \end{pmatrix}. \tag{5.77}$$

Abbildung 5.17 zeigt die Auswirkungen grafisch. Die Messungen streuen in den nicht begehbaren Bereich, die geschätzte Position muss sich aber im begehbaren Bereich vor der Wand befinden. Es ist deutlich zu sehen, dass die Genauigkeit entlang der Nord-Achse höher ist, wenn Ungleichungen aus dem BSP Verfahren berücksichtigt werden. Ebenfalls ist zu sehen, wie die Ausrichtung der Wand einen Einfluss auf die Form und Ausrichtung der BSP-Konfidenzellipse hat. Die Ungleichungen wirken sich hier positiv auf die geschätzte Genauigkeit aus. Das Ergebnis wurde verifiziert mit der M-Schätzer Fehlerfortpflanzung nach Jäger et al. (2005) sowie dem QR Verfahren nach Gleichung 5.51, mit jeweils dem gleichen Ergebnis.

Allgemein stößt man mit Ungleichungen jedoch an die Grenzen der Möglichkeiten mit einer Kovarianzmatrix die a posteriori Wahrscheinlichkeitsdichte zu beschreiben. Das Einschränken der Lösungsmenge durch eine beliebige Raumgeometrie stellt klar eine nichtlineare Transformation dar, die Normalverteilung bleibt jedoch streng nur bei linearen Transformationen erhalten. Bei vielen nichtlinearen Transformationen lässt sich die Verteilung einer zu schätzenden Größe aber dennoch gut über eine Kovarianzmatrix beschreiben; darauf basieren Verfahren wie z. B. das EKF oder auch das Sigma-Point Kalman-Filter. Daher kann, genau wie beispielsweise bei einem EKF, keine Garantie gegeben werden, ob für einen Anwendungsfall das Verfahren gut geeignet ist.

Die tatsächliche Wahrscheinlichkeitsdichtefunktion für das gezeigte Beispiel wurde durch eine Monte-Carlo Simulation bestimmt. Die Grundlagen der Zustandsschätzung auf Bayes-Basis (vgl. Kapitel 2.1) helfen hier, einen tieferen Einblick in die nichtlinearen Zusammenhänge zu bekommen. In $n = 10000$ Durchläufen wurden auf die Messungen jeweils normalverteilte Fehler hinzuaddiert. Dies führt zu Variationen in dem Lösungsvektor $\mathbf{y}$. Diese Variationen sind in Abbildung 5.18 zu sehen. Aufgrund der strengen Bedingungsungleichungen ist kein Lösungsvektor im nicht begehbaren Bereich zu finden.

Die über den Erwartungswert berechnete Position ist in Abbildung 5.18 eingezeichnet. Der Erwartungswert berechnet sich aus der jeweiligen Lösung, multipliziert mit der Wahrscheinlichkeit für diese Lösung:

$$E[\mathbf{y}] = \int \mathbf{y} \cdot p(\mathbf{y}) \, \mathrm{d}y. \tag{5.78}$$

Die Wahrscheinlichkeitsdichtefunktion $p(\mathbf{y})$ kann in einer Monte Carlo Simulation über eine diskrete Auswertung erfolgen. Es wird ein Raster über die verschiedenen Ausprägungen gelegt, anschließend wird gezählt, wie oft ein Lösungsvektor in dieses Raster gelangt und schlussendlich normiert. Abbildung 5.20 und 5.22 zeigen die Wahrscheinlichkeitsdichtefunktion (PDF) für das in Abbildung 5.18 gezeigte Beispiel, also unter Berücksichtigung der Wandungleichungen (BSP). Zum Vergleich zeigen Abbildung 5.19 und 5.21 die Wahrscheinlichkeitsdichtefunktion ohne Wandungleichungen. Ohne Ungleichungen ist die Verteilung flacher, also die Schätzung ist weniger genau.

Jedes Rasterelement stellt eine diskrete Wahrscheinlichkeit dar. Die Summe dieser dargestellten diskreten Wahrscheinlichkeiten für jedes Rasterelement i ergibt genau 1. Vergleichbar mit Gl. 2.2:

$$\sum_{i=1}^{n} p(\mathbf{y}_i) = 1. \tag{5.79}$$

Somit kann über die approximierte Wahrscheinlichkeitsdichtefunktion der Erwartungswert in Gl. 5.78 berechnet werden. Die Konfidenzellipse in Abb. 5.18 wird aus der Kovarianzmatrix ermittelt (Niemeier, 2008):

$$\mathbf{Q}_{yy} = E\big[\, (\mathbf{y} - E[\mathbf{y}]) \cdot (\mathbf{y} - E[\mathbf{y}])^T \,\big]. \tag{5.80}$$

Für die empirische Kovarianz aus den n-Ausprägungen in der Monte-Carlo Simulation mit den Verbesserungen $\mathbf{v}$ gilt (Jäger et al., 2005, S. 64):

$$\hat{\mathbf{Q}}_{yy} = \frac{1}{n-1} \cdot \left[\mathbf{v}\mathbf{v}^T\right] \tag{5.81}$$

$$= \frac{1}{n-1} \cdot \begin{pmatrix} \sum_{i=1}^{n} v_{1i}^2 & \sum_{i=1}^{n} v_{1i} \cdot v_{2i} \\ \sum_{i=1}^{n} v_{2i} \cdot v_{1i} & \sum_{i=1}^{n} v_{2i} \end{pmatrix}. \tag{5.82}$$

Vergleicht man die Konfidenzellipsen aus Abb. 5.17 mit der 50 % Konfidenzellipse aus Abb. 5.18, dann sieht man deutliche Unterschiede. Beide Abbildungen haben den gleichen Maßstab. Die Kovarianzmatrizen aus Abb. 5.17 sind also tendenziell zu pessimistisch. Dafür stellen sie dennoch eine gute Approximation dar.

Zuletzt soll noch auf einen besonderen Aspekt hingewiesen werden. Die Beziehung zwischen der Geometrie und der Kovarianzmatrix wird über die Residuen hergestellt. Die Fehlerfortpflanzung nach Gl. 5.73 ist sensitiv gegenüber Variantionen in den Residuen. Dies hat sich im Rahmen der Monte-Carlo-Simulation gezeigt. Abbildung 5.23 zeigt exemplarisch die Variation der geschätzten Kovarianzmatrix $\mathbf{Q}_{yy}$ des Zustandsvektors $\mathbf{y}$ bei normalverteilten Messungen (hier 6 Abbildungen aus $n = 100$ Durchläufen). Fallen viele der normalverteilen Messungen in den Bereich, der durch Ungleichungen ausgeschlossen ist (also hinter die Wand), so verkleinert sich die Kovarianzmatrix (z. B. Abb. 5.23b). Fallen die Messungen überwiegend vor die Wand, so entspricht die Kovarianzmatrix mit Ungleichungen (*BSP*) der Kovarianzmatrix ohne Ungleichungen (*No BSP*). (Abbildung 5.23c). Die BSP-Kovarianzmatrix ist aber maximal so groß wie die Nicht-BSP Kovarianzmatrix (also bei gleicher Datengrundlage, nur ohne Ungleichungen).

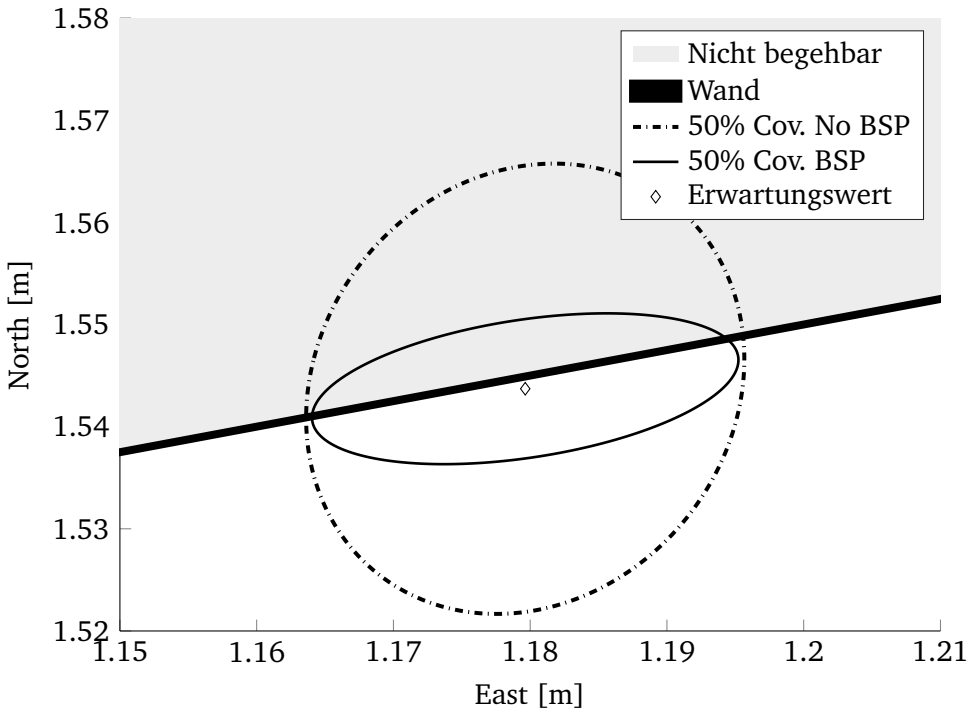


Abbildung 5.17.: Vergleich der Konfidenzellipsen. Die Konfidenzellipse ohne Ungleichungen (No BSP) gegenüber der Konfidenzellipse mit Ungleichungen (BSP).

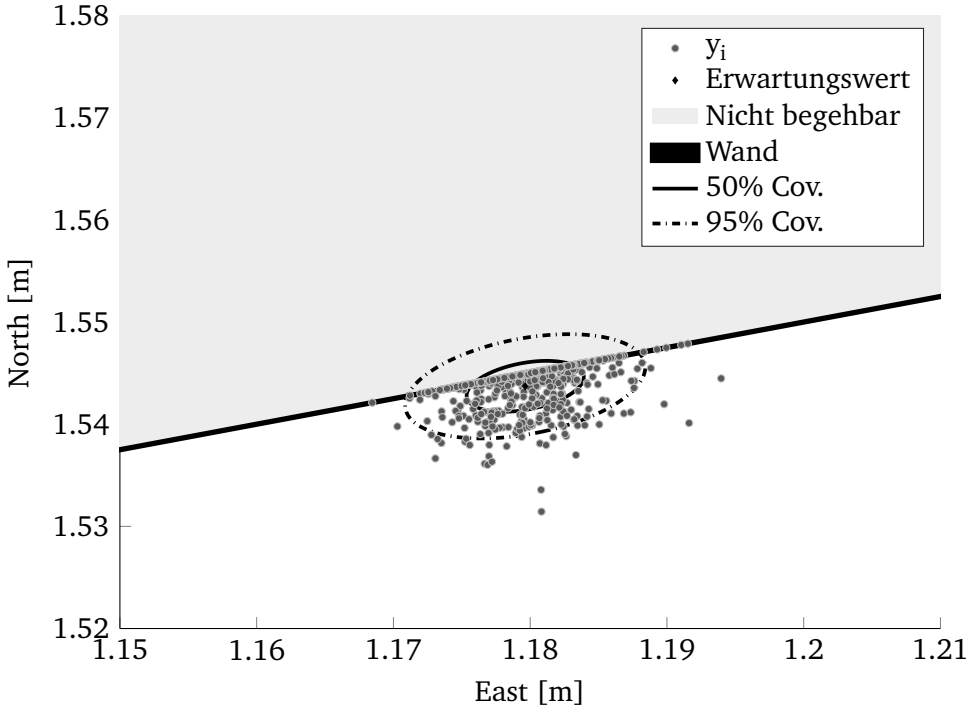


Abbildung 5.18.: Bestimmung der Kovarianzmatrix $\mathbf{Q}_{yy}$ über ein Monte Carlo Verfahren (10000 Durchläufe)

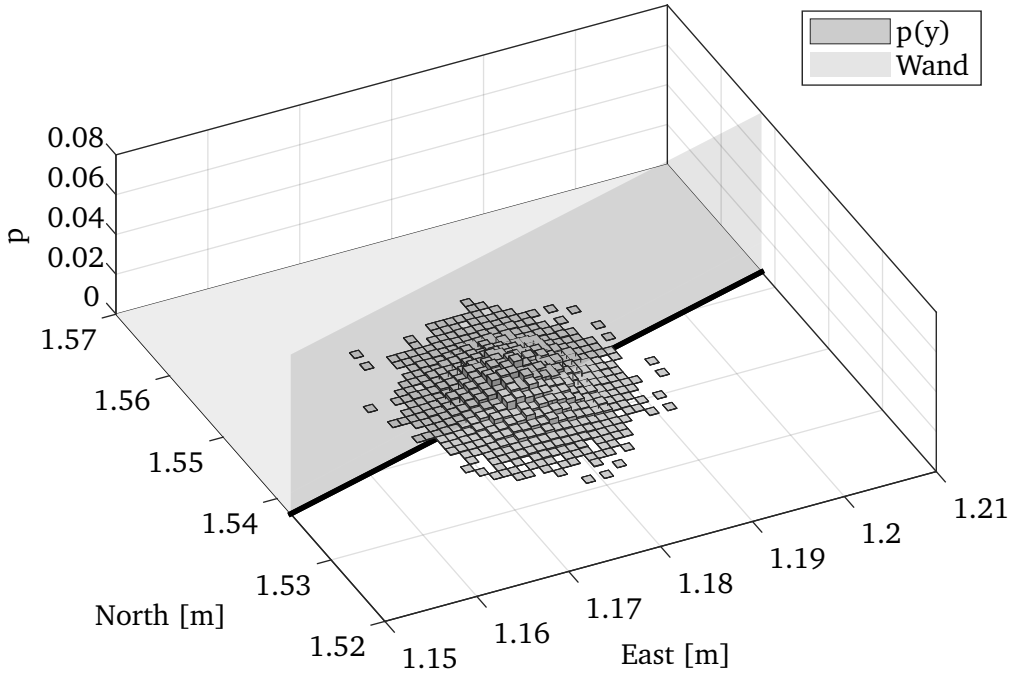


Abbildung 5.19.: 3D Darstellung der Wahrscheinlichkeitsdichtefunktion (PDF) ohne BSP Einschränkungen aus einer Monte-Carlo Simulation (2D Darstellung in Abb. 5.21).

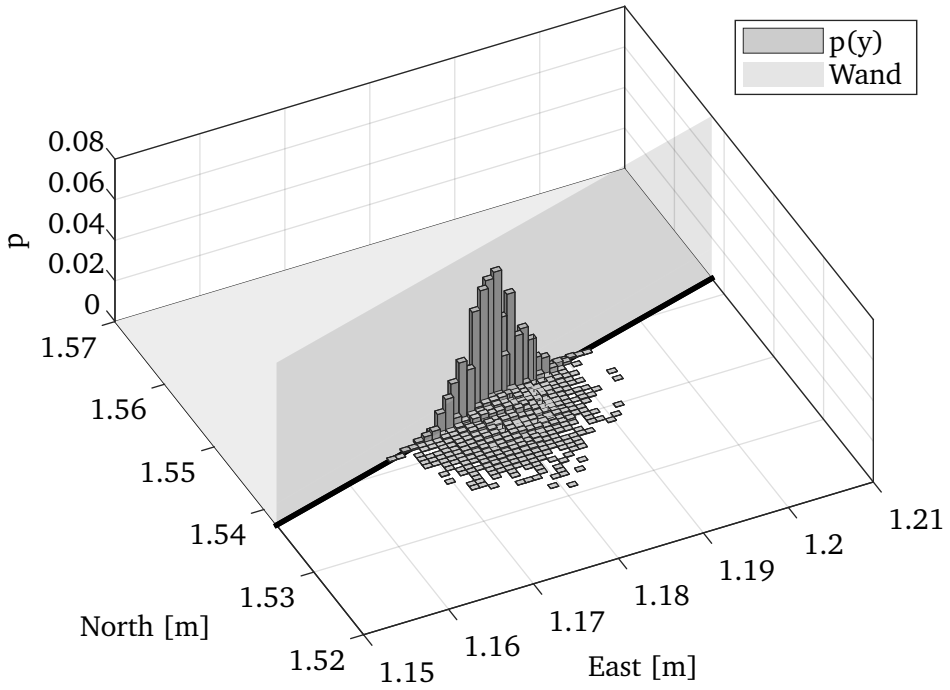


Abbildung 5.20.: 3D Darstellung der Wahrscheinlichkeitsdichtefunktion (PDF) aus einer Monte-Carlo Simulation (2D Darstellung in Abb. 5.22).

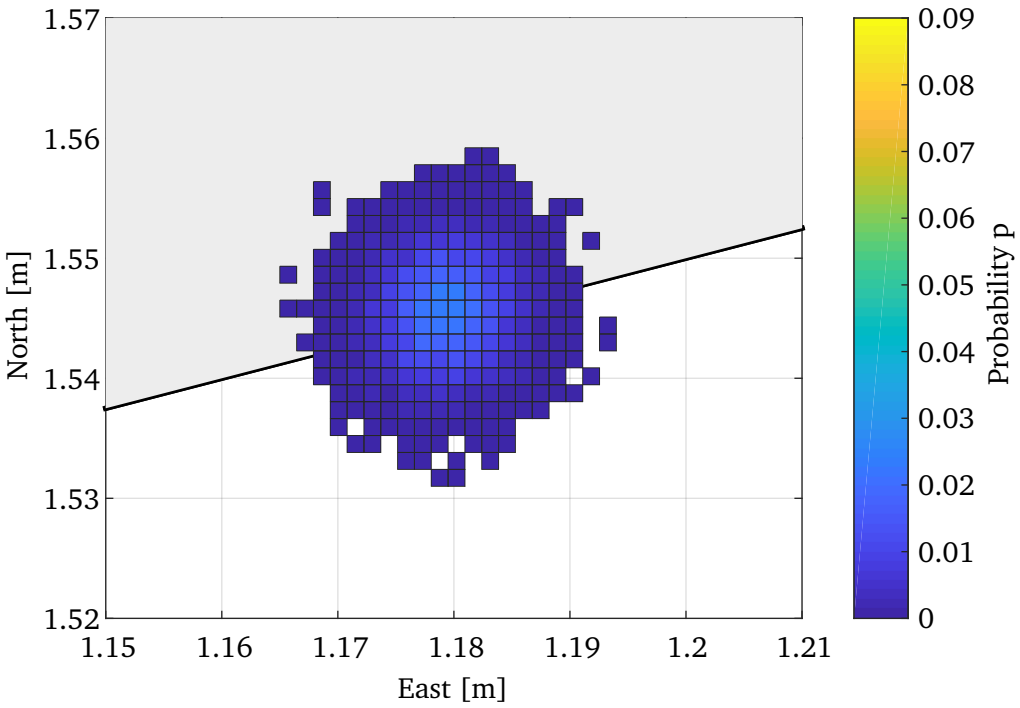


Abbildung 5.21.: 2D Darstellung der Wahrscheinlichkeitsdichtefunktion (PDF) ohne BSP Einschränkungen aus einer Monte-Carlo Simulation (3D Darstellung in Abb. 5.19).

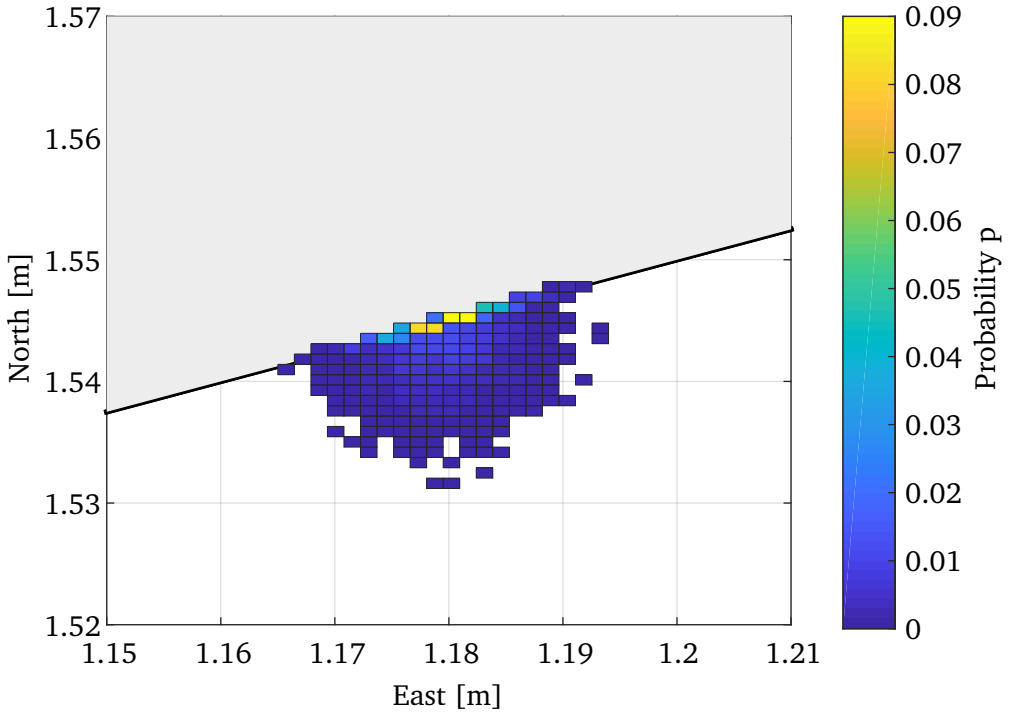


Abbildung 5.22.: 2D Darstellung der Wahrscheinlichkeitsdichtefunktion (PDF) aus einer Monte-Carlo Simulation (3D Darstellung in Abb. 5.20).

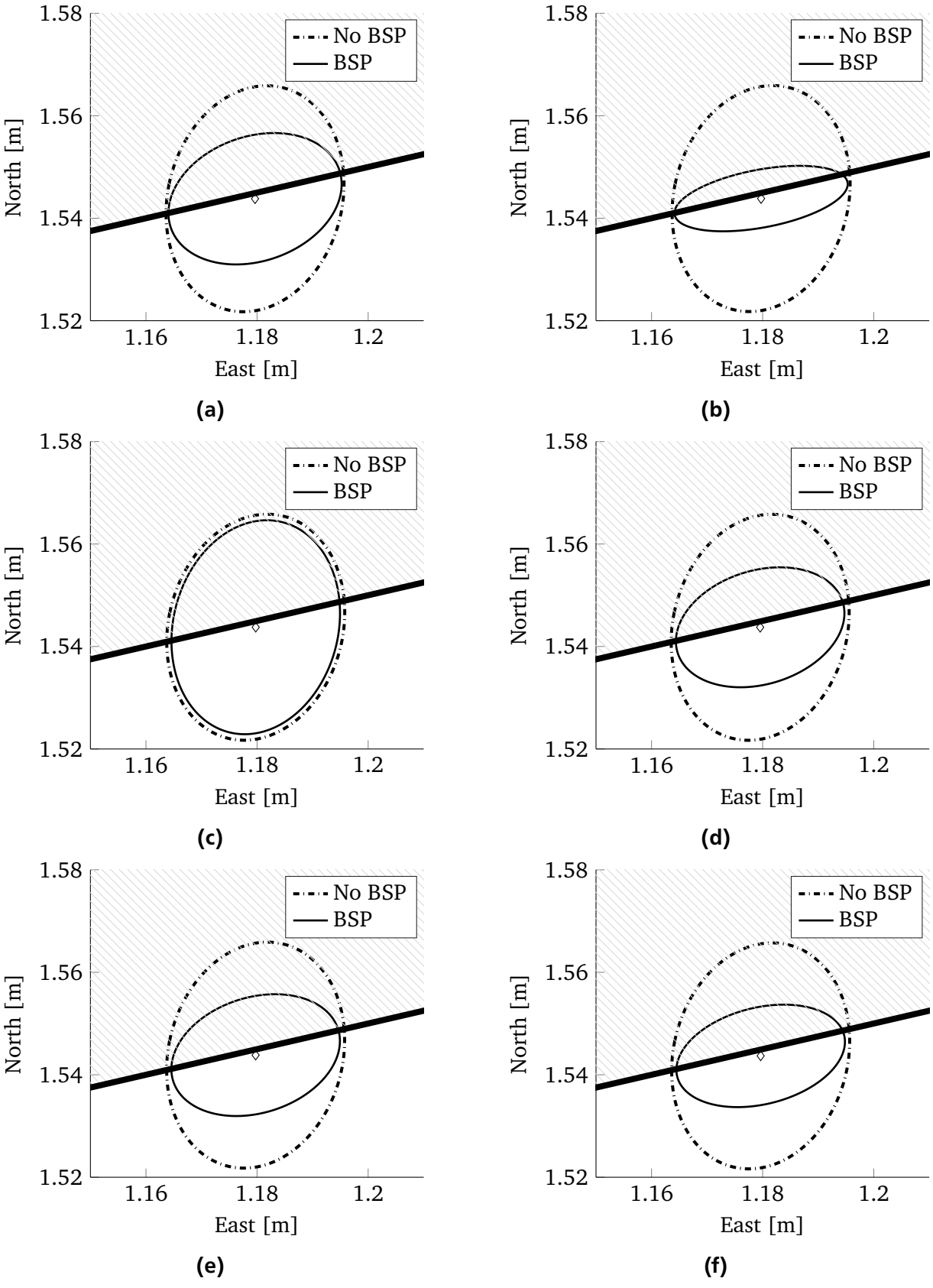


Abbildung 5.23.: Kovarianzsensitivität in der BSP Auswertung.

5.4 Auswertung von Realdaten und synthetischen Daten

Auf Basis der theoretischen Grundlagen in den vorhergehenden Abschnitten in Kapitel 5, besonders Abschnitt 5.2, 5.1.6, sowie 5.1.7 wurde eine neue Methode zur Navigationszustandsschätzung umgesetzt. Das Verfahren baut zudem auf den state-of-the-art Verfahren der Navigation auf, beschrieben in Kapitel 4.

Die für diese Arbeit umgesetzte Implementierung hat den Namen *xfilter* (für Simplex Kalman-Filter) und wurde in den folgenden Programmiersprachen geschrieben: MATLAB, C und C++ Code. Die High-Level Funktionen sind in MATLAB geschrieben, während rechenintensive Subsysteme in C/C++ umgesetzt wurden. Der Name *xfilter* ist an das sogenannte *qfilter* angelehnt und baut auf diesem auf. Die *qfilter* Umgebung ist eine Anwendungsplattform zur Navigation, die der Autor im Rahmen des *Navka* Projekts entwickelt hat. Somit basiert das *xfilter* auf bewährten Algorithmen, die in verschiedenen Anwendungsgebieten erprobt und verifiziert wurden, wie z. B. der Navigation von Fluggeräten (siehe Abschnitt 6), Pedestrian-Navigation, Fahrzeug-Navigation und auch Bird-Tracking. Siehe dazu auch Zwiener (2012), Jäger et al. (2013a), Zwiener et al. (2014), Zwiener et al. (2015), u. Jäger und Zwiener (2016).

Das *xfilter* besteht im Kern aus dem Simplex Kalman-Filter Kern, der in Abschnitt 5.2 beschrieben wird, zudem ist die Umsetzung in MATLAB u. C/C++ in Anhang A und B zu finden. Die Software hat eine Schnittstelle zum Einlesen von IMU Daten (Navka SIMA Datenformat) u. GNSS Rohdaten. Für die Verarbeitung der GNSS Rohdaten wird auf die Open-Source RTKLIB Bibliothek aufgebaut[4] (Takasu und Yasuda, 2009). Folgende Funktionen der RTKLIB werden verwendet und unterstützen verschiedene Satellitennavigationssysteme, dadurch konnte mit relativ wenig Aufwand GLONASS in das *xfilter* eingebunden werden (Galileo oder Beidou wären ebenfalls integrierbar):

- SATPOSS(...): Berechnung der Satelliten Position $\mathbf{r}^s$ in Gl. 4.23, inklusive der Korrektur der Satellitenuhrenfehler (t^s) aus Gl. 4.22
- GEODIST(...): Berechnung des Vektors $\mathbf{e}_k$ in Gl. 4.24 inklusive der Sagnac Korrektur, siehe Gl. 3.59
- TROPCORR(...): Berechnung der Troposphärenkorrektur ΔT in Gl. 4.22
- IONOCORR(...): Berechnung der Ionosphärenkorrektur ΔI in Gl. 4.22

5.4.1 Testfahrt Karlsruhe

Verschiedene Testfahrten wurden in Karlsruhe durchgeführt mit dem Ziel, Rohdaten von Low-Cost Sensoren zu sammeln (IMU, Magnetometer, Barometer, GNSS). Der Testaufbau besteht aus den Low-Cost Sensoren in Tabelle 5.1. Der Navigationscomputer FC4 (Abb. 5.24) des *Navka* Projekts ist eine Eigenentwicklung, entwickelt an der Hochschule Karlsruhe (HSKA). Die in Abschnitt 4 beschriebenen Verfahren sind direkt auf dem verbauten Mikrocontroller[5] lauffähig. Das Simplex Kalman-Filter Verfahren wurde aber nicht auf diesen Mikrocontroller portiert, sodass die FC4 Einheit hier nur zum Erfassen von kalibrierten Beobachtungen dient. Die Navka FC4 Einheit kann darüber hinaus als Flugregelungscomputer

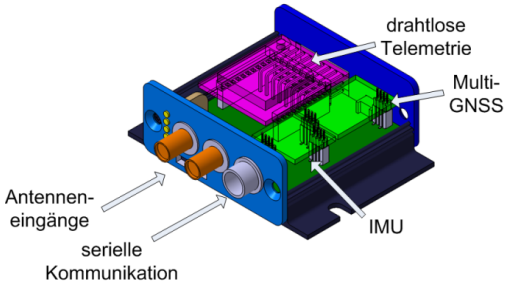
[4] Online unter `http://www.rtklib.com/` sowie `https://github.com/tomojitakasu/RTKLIB`, in dieser Arbeit wurde der Stand vom *2016/01/26 Version 2.4.3* verwendet.

[5] STM32F407 168 MHz ARM Cortex M4 MCU.

Tabelle 5.1.: Testaufbau Testfahrten Karlsruhe.

Sensor	Modell
IMU	InvenSense MPU9150 (Gyroskop + Accelerometer)
Barometer	MEAS Switzerland MS5611-01
GNSS Receiver	u-blox M8T Single Frequency (L1)
Low-Level Datenerfassung	Navka Navigationscomputer FC4 + Laptop
Magnetometer	AKM AK8975

arbeiten, dieser Modus bildet die Grundlage für die in Kapitel 6 vorgestellten Themen. Abbildung 5.24c zeigt rechts unten die PWM-Anschlüsse zur Motoransteuerung (Pulsweitenmodulation). Abbildung 5.25c



(a) Navka FC4 CAD Zeichnung.

(b) Navka FC4 mit Aluminiumgehäuse.

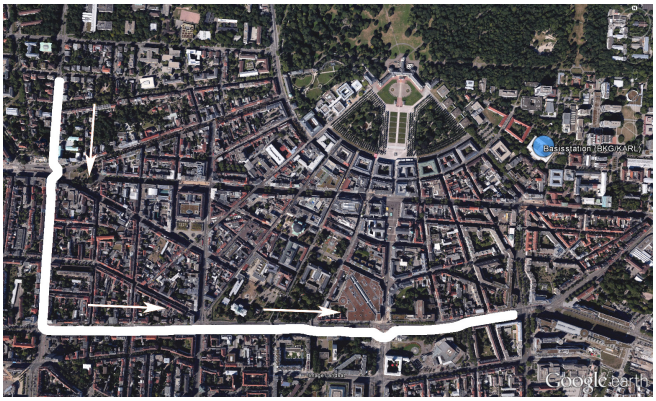
(c) Navka FC4 Platinenansicht.

Abbildung 5.24.: Navka FC4 Navigations- und Flugregelungscomputer.

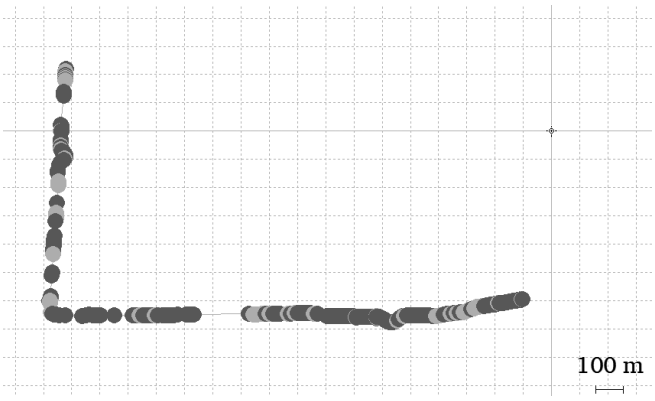
zeigt die Installation in einem Fahrzeug. Die Länge des GNSS Antennen Leverarms liegt bei 0,96 m, die Bestimmung des Leverarms kann direkt aus Abbildung 5.25c mit den markierten *AprilTags* (Olson, 2011) erfolgen[6]. Die verzögerte Verfügbarkeit der GNSS Messungen wurde korrigiert, siehe Abschnitt 4.4.

Eine Testfahrt vom 28.7.2015 in Karlsruhe soll im Folgenden exemplarisch analysiert werden. Der Verlauf ist in Abbildung 5.25a dargestellt, der Startpunkt liegt im Nordwesten, die Fahrt geht in südlicher Richtung, bis dann in Richtung Osten weitergefahren wird. Es wurden im Monat Juli 2015 mehrere Testfahrten durchgeführt, keine der Testfahrten hatte signifikante Ausreißer bei den GNSS Beobachtungen. Die Signalqualität der aktuellen Empfängergeneration ist selbst bei Low-Cost Geräten gut; hinzukommt, dass Karlsruhe meistens nicht von sehr hohen Gebäuden geprägt ist, die zu Multipatheffekten beitragen

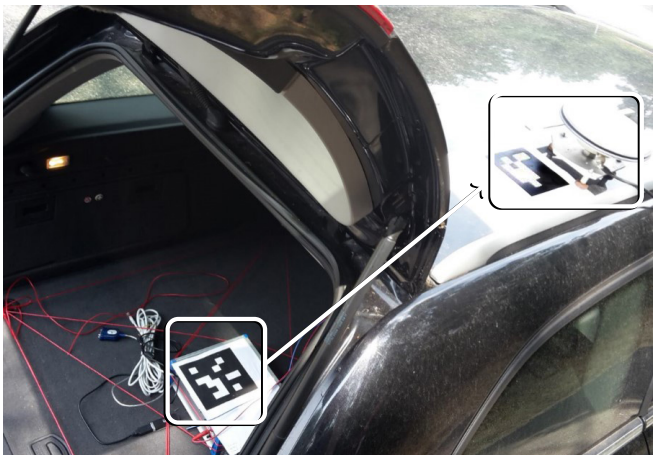
[6] Bei bekannter Druckgröße sowie zusätzlich noch mit einer manuellen Korrektur zwischen den Markern und dem IMU Zentrum sowie dem Antennenphasenzentrum.



(a) Testfahrt in Karlsruhe (Bild: Google Earth).



(b) GNSS Verfügbarkeit.



(c) Aufbau der Sensorik in einem Fahrzeug.

(d) Google Earth 3D Ansicht.

Abbildung 5.25.: Testaufbau für Messungen in Karlsruhe.

können. Von einem „Urban Canyon“ kann nicht gesprochen werden, siehe Abb. 5.25d. Eine durchgehende Verfügbarkeit einer reinen GNSS Lösung ist dennoch nicht ständig gegeben. Brücken und Tunnel führen temporär zu GNSS Aussetzern, siehe Abbildung 5.25b. Bei einer gültigen GNSS Lösung ist ein Punkt eingezeichnet. An den Lücken sieht man deutlich, dass die Inertialnavigation notwendig ist, um kurzzeitige GNSS Abschattungen zu überbrücken. Da nun aber ein im Kern auf Robustheit ausgelegter Zustandsschätzalgorithmus getestet werden soll, wurden für eine Zeitdauer von 15 Sekunden 4 Satelliten manuell mit grob falschen L1-Frequenz Pseudoranges verfälscht, bei insgesamt 12 sichtbaren Satelliten (GPS+GLONASS). Die Fehler in den L1 Pseudoranges liegen bei 40 m, 60 m, 80 m und 100 m. Der Vorteil bei diesem Ansatz ist auch, dass die Fehler klar bekannt sind und keine Unbekannte in der Auswertung darstellen. Zudem ist gerade der maximale Einfluss des Fehlers interessant, vgl. Abschnitt 2.5. Besonders auch im Hinblick auf den möglichen Einsatz in einem Fluggerät, wo selbst bei unwahrscheinlichen Fällen eine brauchbare Zustandsschätzung benötigt wird. Die Daten wurden nun mit zwei verschiedenen Ansätzen verarbeitet um zu einer Navigationszustandsschätzung zu kommen:

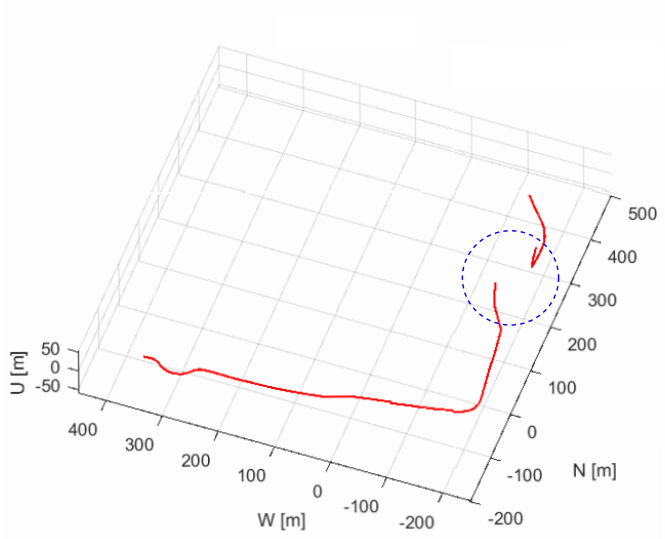
- a) *qfilter*: State-of-the-Art $\mathscr{L}_2$ Error-State Verfahren (beschrieben in Kapitel 4)
- b) *xfilter*: neues Simplex $\mathscr{L}_1$ Kalman-Filter Verfahren (beschrieben in Kapitel 5.2)

In Abbildung 5.26 werden beide Methoden gegenübergestellt. Dieselben Daten wurden im Post-Processing mit Methode a) (linke Spalte) sowie Methode b) (rechte Spalte) verarbeitet. In den Ein-

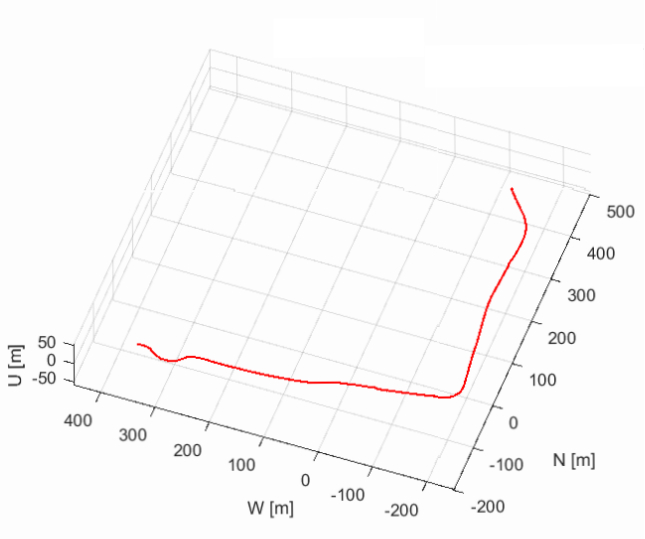
gangsdaten wurden zwischen Sekunde 8600 und 8615 (Zeitstempel) die L1 Pseudoranges verfälscht. In der 3D Trajektorienansicht ist bei der $\mathscr{L}_2$ Schätzung in Abb. 5.26a deutlich sichtbar, wie die Zustandsschätzung durch die hohen Messfehler völlig unbrauchbar wird. Das ist ohne weitere Gegenmaßnahmen so auch zu erwarten. Bei der $\mathscr{L}_1$ Simplex Kalman-Filter Zustandsschätzung hingegen ist durchgängig eine verwertbare Positionslösung vorhanden, siehe Abb. 5.26b. Der geschätzte Empfängeruhrenfehler ist in Abb. 5.26c und Abb. 5.26d zu sehen. Der Uhrenfehler wurde in eine Meterangabe umgerechnet, also dem Fehleranteil des Uhrenfehlers auf der Pseudorangemessung. Hier ist praktisch kein Einfluss auf die $\mathscr{L}_1$ Schätzung zu erkennen. Beide Ansätze schätzen bis zu Sekunde 8600 in etwa den gleichen Gyroskop Bias, was durchaus ein interessanter Aspekt ist, da die $\mathscr{L}_1$ Simplex Kalman-Filter Methode damit plausibilisiert wird. Ab Sekunde 8600 verfälschen die Pseudorangefehler im $\mathscr{L}_2$ Fall die Bias-Schätzung stark. Beispielsweise über $10°/s$ entlang der Y-Achse, was sofort zu massiven Lagefehlern führt. Auch hat das $\mathscr{L}_2$ Filter nach der Fehlerperiode Schwierigkeiten sich zu „*erholen*“ und den ursprünglichen Bias wieder zu schätzen. Tabelle 5.2 stellt die quantifizierten Ergebnisse der Testfahrt vom 28.7.2015 dar. Dabei wird der Abschnitt ohne Ausreißer getrennt im unteren Teil von Tabelle 5.2 aufgeführt, um das $\mathscr{L}_1$ Verfahren dem $\mathscr{L}_2$ Verfahren auch unter normalen Bedingungen gegenüberzustellen. Bemerkenswert ist, dass die Position hier auch ohne groben Fehler bei dem $\mathscr{L}_1$ Verfahren genauer ist (5.91 m gegen 7.00 m).

Tabelle 5.2.: Auswertung Testfahrt 28.7.2015.

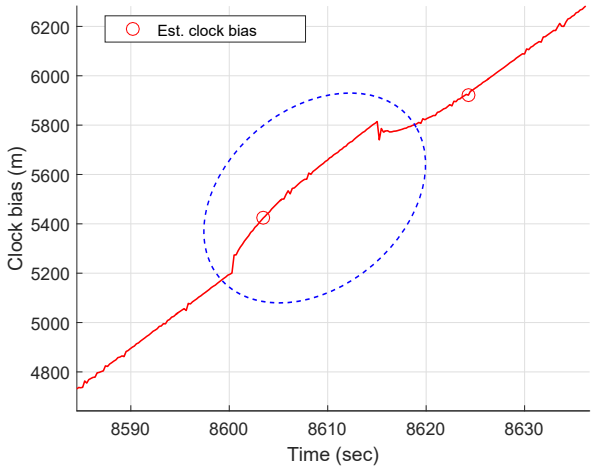
Genauigkeit	**Einheit**	$\mathscr{L}_2$	$\mathscr{L}_1$
Position	m	24.92	5.02
Roll-Winkel ϕ $(1\,\sigma)$	°	0.26	0.33
Pitch-Winkel θ $(1\,\sigma)$	°	0.26	0.35
Yaw-Winkel ψ $(1\,\sigma)$	°	1.17	1.43
Geschwindigkeit $(1\,\sigma)$	m/s	0.04 (N)	0.13 (N)
		0.03 (E)	0.10 (E)
		0.03 (D)	0.10 (D)
Acc. Bias XYZ $(1\,\sigma)$	m/s^2	0.03 (X)	0.04 (X)
		0.04 (Y)	0.04 (Y)
		0.006 (Z)	0.006 (Z)
Gyro. Bias XYZ $(1\,\sigma)$	°/s	0.0004 (X)	0.0004 (X)
		0.0004 (Y)	0.0004 (Y)
		0.0008 (Z)	0.0008 (Z)
Position	m	7.00	5.91
Roll-Winkel ϕ $(1\,\sigma)$	°	0.33	0.42
Pitch-Winkel θ $(1\,\sigma)$	°	0.30	0.40
Yaw-Winkel ψ $(1\,\sigma)$	°	1.37	1.79
Geschwindigkeit $(1\,\sigma)$	m/s	0.04 (N)	0.11 (N)
		0.03 (E)	0.11 (E)
		0.03 (D)	0.11 (D)
Acc. Bias XYZ $(1\,\sigma)$	m/s^2	0.045 (X)	0.057 (X)
		0.045 (Y)	0.060 (Y)
		0.0049 (Z)	0.0062 (Z)
Gyro. Bias XYZ $(1\,\sigma)$	°/s	0.0003 (X)	0.0004 (X)
		0.0003 (Y)	0.0004 (Y)
		0.0009 (Z)	0.0010 (Z)



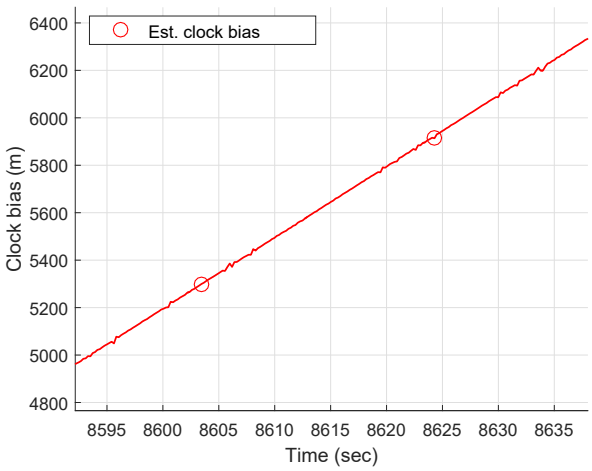
(a) $\mathscr{L}_2$ Kalman-Filter.



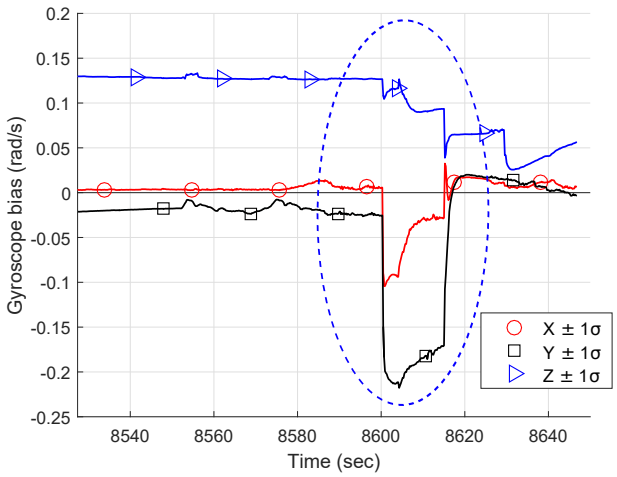
(b) $\mathscr{L}_1$ Simplex Kalman-Filter.



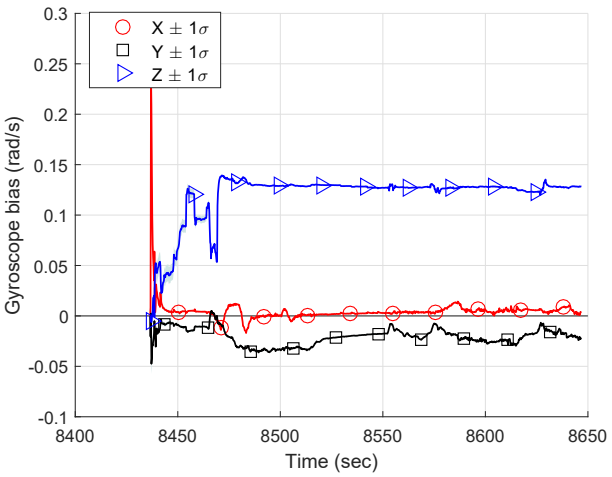
(c) $\mathscr{L}_2$ GNSS Uhrenfehler Schätzung.



(d) $\mathscr{L}_1$ Simplex GNSS Uhrenfehler Schätzung.



(e) $\mathscr{L}_2$ Gyroskop Bias Schätzung.

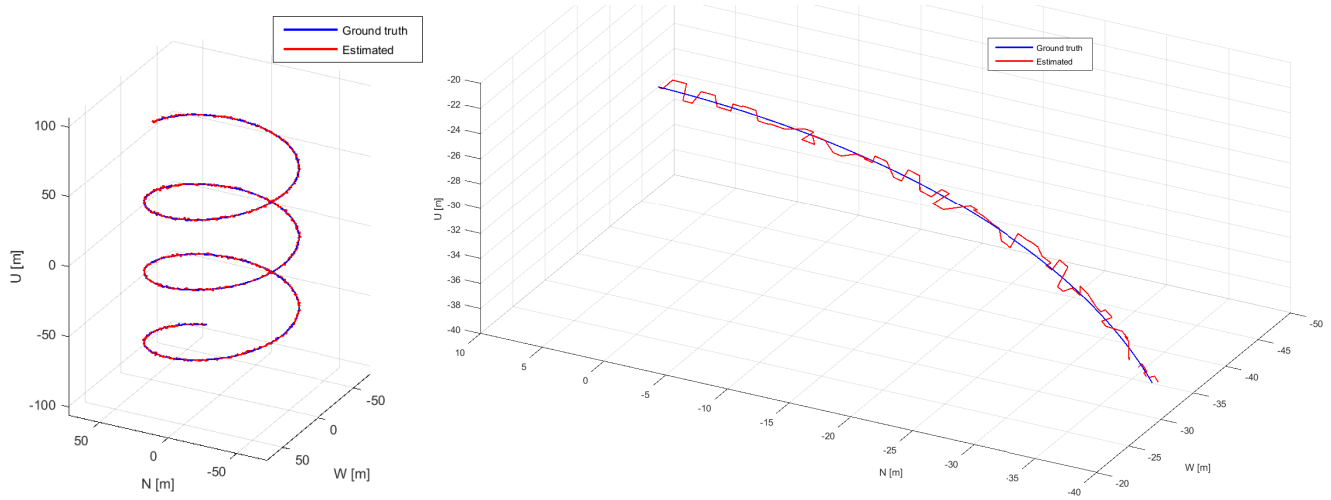


(f) $\mathscr{L}_1$ Simplex Gyroskop Bias Schätzung.

Abbildung 5.26.: Vergleich zwischen Methode a) $\mathscr{L}_2$ Error-State Kalman-Filter und b) $\mathscr{L}_1$ Simplex Kalman-Filter. Grobe Fehler zwischen Sekunde 8600 u. 8615 induziert. Störeinflüsse in diesem Bereich sind durch eine gestrichelte Ellipse hervorgehoben.

5.4.2 Synthetische Beobachtungen

Während im vorhergehenden Abschnitt das $\mathscr{L}_1$ Simplex Kalman-Filter mit Realdaten getestet wurde, bilden in diesem Abschnitt synthetische Daten aus der Simulationsumgebung SIMA die Grundlage. Dabei soll kritisch die Performance des Verfahrens im Detail diskutiert werden. Abbildung 5.27a zeigt eine Simplex Kalman-Filter Positionsschätzung gegenüber der stetigen Helix Trajektorie. Das Simplex Kalman-Filter bekommt als Eingabedaten IMU Messungen sowie L1 GNSS Pseudoranges. Die von SIMA simulierten Pseudorangefehler δPSR haben eine Verteilung von $\delta\text{PSR} \sim \mathcal{N}(0, 0.4)$ in Metern. In Abbildung 5.27b wird ein Ausschnitt aus Abbildung 5.27a gezeigt, hier sieht man deutlich, wie unstetig die geschätzte $\mathscr{L}_1$ Lösung ist. Das ist auf die Interpolationscharakteristik der $\mathscr{L}_1$ Schätzung zurückzuführen (siehe Abschnitt 5.1.6), also der Eigenschaft, dass der Lösungsvektor die Beobachtungen berührt. Deutlich wird dies weiter, wenn man die Standard-Abweichungen der $\mathscr{L}_2$ Lösung mit der $\mathscr{L}_1$ Lösung

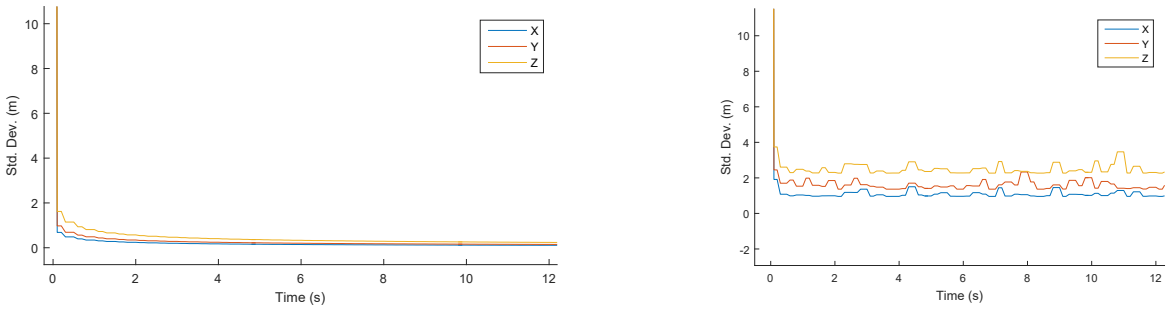


(a) SIMA Helix Trajektorie.

(b) Ausschnitt aus der Helix Trajektorie.

Abbildung 5.27.: Simplex $\mathscr{L}_1$ Navigationszustandsschätzung entlang einer Nominaltrajektorie in einem North-West-Up (NWU) Koordinatensystem.

vergleicht. Die $\mathscr{L}_1$ Lösung erreicht nicht die Genauigkeit der $\mathscr{L}_2$ Lösung, was auch zu erwarten ist. Diese Einschränkung des vorgestellten Verfahrens soll aber damit hervorgehoben werden. Für Anwendungs-



(a) $\mathscr{L}_2$ Std.-Abweichung der Position.

(b) $\mathscr{L}_1$ Std. Abweichung der Position.

Abbildung 5.28.: Geschätzte innere Genauigkeit der Positionslösung.

fälle, in denen Post-Processing möglich ist (was in der Navigation und Regelung von Luftfahrtgeräten

i. A. nicht der Fall ist), kann ein Glättungsfilter (engl. *optimal smoother*) eingesetzt werden. Sehr häufig wird ein Rauch-Tung-Striebel-Smoother genutzt, ein sogenannter Fixed-Interval-Smoother (Rauch et al., 1965). Dieses Verfahren kann die Genauigkeit signifikant erhöhen, da für jede Epoche k Informationen aus der Zukunft genutzt werden. Ein RTS-Smoother setzt an, nachdem das Kalman-Filter Verfahren (oder eben Simplex Kalman-Filter Verfahren) wie gehabt für alle Epoche $k = 0 \dots N$ durchgeführt wurde. Dabei müssen die Kalman-Filter Zustände (a priori u. a-posteriori Zustände $\mathbf{y}_k^-, \mathbf{y}_k^+$ inkl. der Kovarianzmatrizen $\mathbf{Q}_k^+, \mathbf{Q}_k^-$) gespeichert werden. Rückwärtsgerichtet spielt es dann keine Rolle, ob mit der $\mathscr{L}_1$ oder der $\mathscr{L}_2$ Norm gerechnet wurde, da hier die Wahrscheinlichkeiten in den Kovarianzmatrizen betrachtet werden:

$$\mathbf{K}_k = \mathbf{Q}_k^+ \cdot \mathbf{\Phi}_k^T \cdot \left(\mathbf{Q}_{k+1}^-\right) \tag{5.83}$$

$$\mathbf{y}_{k,\text{smooth}} = \mathbf{y}_k^+ + \mathbf{K}_k \cdot \left(\mathbf{y}_{k+1,\text{smooth}} - \mathbf{y}_{k+1}^-\right). \tag{5.84}$$

Der RTS-Smoother setzt in der letzten Epoche N mit dem vorwärtsgerichteten a priori Zustandsvektor an: $\mathbf{y}_{N,\text{smooth}} = \mathbf{y}_N^+$. An dieser Stelle soll das Verfahren nicht näher betrachtet werden, es sei auf die Standardwerke verwiesen: Gelb (1974), Wendel (2011). In (Becker, 2016, Kap. 3.6) werden zudem die RTS-Smoothing Zusammenhänge für Error-State Kalman-Filters dargestellt. Exemplarisch zeigt Abbildung 5.29 den Genauigkeitsgewinn des Verfahrens.

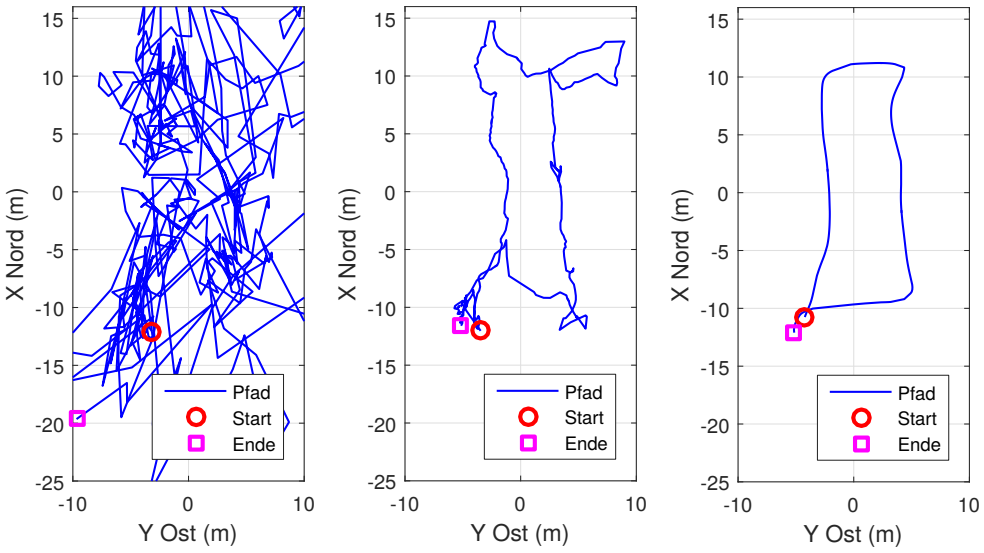


Abbildung 5.29.: Smoothing v.l.n.r.: ungefilterte Positionen, vorwärts gefilterte Positionen, Rauch-Tung-Striebel rückwärts gefilterte Positionen.

Eine Besonderheit des Simplex Kalman-Filter Verfahrens ist, dass es i. A. nicht empfehlenswert ist Messungen sequentiell zu verarbeiten. Bei einem gewöhnlichen Kalman-Filter ist es üblich, dass durch eine Dekorrelation der Messungen (siehe Gl. 2.83) die Verarbeitung sequentiell erfolgt. Dies minimiert den Rechenaufwand und kann auch eine Implementierung der Algorithmen vereinfachen. Die Messungen müssen, salopp gesagt, nicht alle in einen „Topf“ geworfen werden. Anders hingegen verhält es sich bei dem Simplex Kalman-Filter. Wenn in Einzelschritten jeweils der geschätzte a priori Zustandsvektor $\mathbf{y}_k^-$ mit den Beobachtungen über Gleichung 5.52 verarbeitet wird, so kommt der Effekt der *Interpolationscharakteristik der $\mathscr{L}_1$ Schätzung* zum Tragen (Abdelmalek und Malek, 2008). Das Simplex Verfahren

tendiert in diesem Fall dazu Messungen zu ignorieren. Wenn aber, beispielsweise wie in Abschnitt 5.4.1 gezeigt, viele Messungen mit dem Zustandsvektor in einem Schritt in Verbindung gebracht werden, so kann das Verfahren in einem Durchlauf Ausreißer verwerfen.

6 Flugphysik und Regelung von Multirotor-Fluggeräten

6.1 Motivation

Während sich die vorhergehenden Kapitel mit der Anwendung des Simplex Algorithmus für die Navigationszustandsschätzung beschäftigen, soll im Rahmen dieser Arbeit gezeigt werden, dass der Simplex Algorithmus auch gewinnbringend für die Regelung von Multirotor-Fluggeräten genutzt werden kann. Die robusten Eigenschaften der $\mathscr{L}_1$-Norm helfen dabei die Fehlertoleranz zu erhöhen sowie die Ausfallwahrscheinlichkeiten des Gesamtsystems zu reduzieren.

Abschnitt 6.2, 6.3 und 6.4 behandeln die Grundlagen der Flugphysik, Simulation und Regelung von Multirotor-Fluggeräten. Abschnitt 6.6 stellt einen neue Formulierung für die $\mathscr{L}_1$-basierte Motorallokation vor. Abschnitt 6.7 zeigt eine verteilte Systemarchitektur, die den Simplex Algorithmus zur Auswahl der Motorstellgrößen bzw. der Ansteuerung der Aktorik nutzt. In Abschnitt 6.8 wird ein Anwendungsfall betrachtet, der besonders von den robusten Eigenschaften der $\mathscr{L}_1$-Norm profitiert: in Bodennähe sind die barometrischen Messungen von Multirotor-Fluggeräten grob verfälscht und müssen entsprechend behandelt werden.

6.2 Flugphysik

Das dynamische Verhalten der Lage von Multirotor-Fluggeräten wird durch die Starrkörperbewegung (engl. *Rigid-Body-Dynamics*) beschrieben. Zentral ist dabei die Eulersche Kreiselgleichung:

$$\mathbf{m}^b_{\text{Aktorik}} + \mathbf{m}^b_{\text{Extern}} = \boldsymbol{\omega}^b_{ib} \times \mathbf{J}^b \cdot \boldsymbol{\omega}^b_{ib} + \mathbf{J}^b \cdot \dot{\boldsymbol{\omega}}^b_{ib}. \tag{6.1}$$

Die Drehmomente $\mathbf{m}^b_{\text{Aktorik}}$ im körperfesten Koordinatensystem b (Einheit Nm), sowie das Massenträgheitsmoment des Fluggeräts $\mathbf{J}^b$ und die aktuellen Drehraten $\boldsymbol{\omega}^b_{ib}$ (rad/s) haben einen Einfluss auf die Änderung der Drehraten. $\mathbf{m}^b_{\text{Extern}}$ beschreibt die Summe weiterer einwirkender Drehmomente[1]. Für N Motoren mit dem tiefgestellten Index i, verortet an den Punkten $\mathbf{r}^b_i$ (bezogen auf den Masseschwerpunkt) lässt sich das gesamte durch die Motoren erzeugte Drehmoment aufteilen in das Drehmoment, das sich aus dem Schub ($\mathbf{f}^b_i$ in Newton) ergibt, sowie das Drehmoment der Motoren selbst ($\mathbf{m}^b_i$):

$$\mathbf{m}^b_{\text{Aktorik}} = \sum_{i=1}^{N} \mathbf{r}^b_i \times \mathbf{f}^b_i + \sum_{i=1}^{N} \mathbf{m}^b_i. \tag{6.2}$$

[1] Gl. 6.1 kann darüber hinaus als Grundlage dienen, um unbekannte Momente $\mathbf{m}^b_{\text{Extern}}$ zu schätzen.

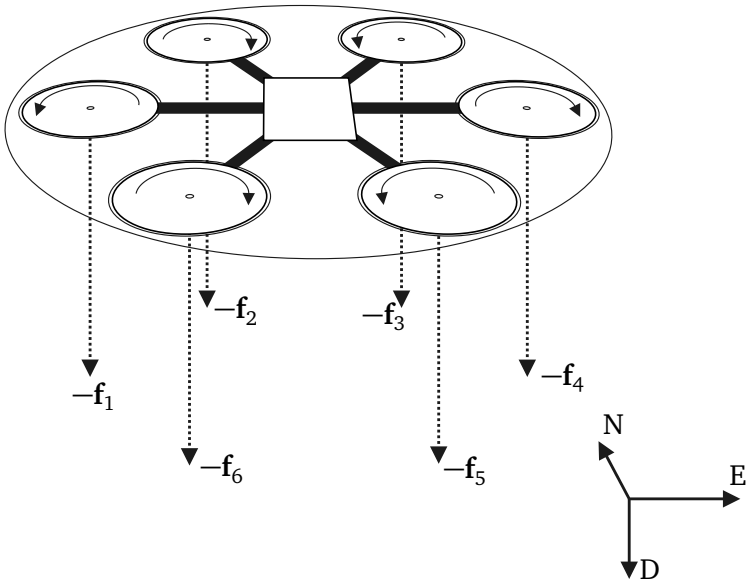


Abbildung 6.1.: Multirotor Beispielkonfiguration mit parallel angeordneten Motoren.

Um die Kreiselmomente der drehenden Propeller/Motoren korrekt zu modellieren, wird zu dem Massenträgheitsmoment des Fluggeräts $\mathbf{J}^b$ noch das 3×3 Massenträgheitsmoment $\mathbf{J}^b_{r_i}$ der N Propeller/Motoren hinzuaddiert (Gl. 6.3). Streng genommen ist damit nicht das Massenträgheitsmoment des Gesamtmotors gemeint, sondern nur der bewegliche Teil des Außenläufers. Die unbewegliche Motormasse wird in $\mathbf{J}^b$ eingerechnet.

$$\mathbf{J} = \mathbf{J}^b + \sum_{i=1}^{N} \mathbf{J}^b_{r_i}. \tag{6.3}$$

Zudem sei die Drehrate des i-ten Propellers relativ zum körperfesten Koordinatensystem des Fluggeräts der 3×1 Vektor $\boldsymbol{\omega}^b_{br_i}$. Diese Drehrate ist im körperfesten Koordinatensystem des Fluggeräts angegeben, für verkippte Motoren muss zuerst in b transformiert werden (siehe Gl. 6.7). Gleichung 6.1 kann dann auf dieser Grundlage nach den Änderungsraten umgestellt werden:

$$\dot{\boldsymbol{\omega}}^b_{ib} = \mathbf{J}^{-1} \cdot \left[\mathbf{m}^b_{\text{Gesamt}} - \left[\boldsymbol{\omega}^b_{ib} \times \right] \cdot \left(\mathbf{J} \cdot \boldsymbol{\omega}^b_{ib} + \sum_{i=1}^{N} \mathbf{J}^b_{r_i} \cdot \boldsymbol{\omega}^b_{br_i} \right) - \sum_{i=1}^{N} \mathbf{J}^b_{r_i} \cdot \dot{\boldsymbol{\omega}}^b_{br_i} \right]. \tag{6.4}$$

Somit liegt mit Gl. 6.4 eine Differentialgleichung vor, die das Änderungsverhalten der Drehraten $\boldsymbol{\omega}^b_{ib}$ und damit auch das zeitliche Verhalten der Fluggerätelage beschreibt.

Für einen idealen symmetrischen Körper können die Nebendiagonalelemente von **J** als 0 angenommen werden:

$$\mathbf{J}^b = \begin{pmatrix} J_x & 0 & 0 \\ 0 & J_y & 0 \\ 0 & 0 & J_z \end{pmatrix}. \tag{6.5}$$

Diese Vereinfachung wird oft genutzt, ganz allgemein kann jedoch die Berechnung des Massenträgheitsmoments als Summation der Punktmassen m_i des Fluggeräts (an den jeweiligen Positionen x_i, y_i und z_i) erfolgen:

$$\mathbf{J}^b = \begin{pmatrix} \sum_i m_i \left(y_i^2 + z_i^2\right) & -\sum_i m_i x_i y_i & -\sum_i m_i x_i z_i \\ -\sum_i m_i x_i y_i & \sum_i m_i \left(x_i^2 + z_i^2\right) & -\sum_i m_i y_i z_i \\ -\sum_i m_i x_i z_i & -\sum_i m_i y_i z_i & \sum_i m_i \left(x_i^2 + y_i^2\right) \end{pmatrix}. \tag{6.6}$$

Abbildung 6.1 zeigt exemplarisch eine Multikopterkonfiguration mit sechs Motoren in einem Body-Koordinatensystem. Allgemein wird ein Multirotor-Flüggerät mit starrer Konfiguration der Aktorik über die Änderung der Motorstellgrößen gesteuert. D. h. der gewünschte Navigationszustand wird allein über die Ansteuerung der i. A. starr verbauten Motoren erreicht. Ein Motor i erzeugt in seinem motoreigenen Koordinatensystem eine Kraft f_i (in $\frac{m}{s^2}$) entgegen der abwärts gerichteten Achse (Down-Achse). Wenn der Motor gegenüber dem Body-Koordinatensystem mit dem Neigungswinkel θ und Azimut ψ gedreht ist, lässt sich die Kraft entsprechend transformieren:

$$\mathbf{f}_i^b = -f_i \cdot \begin{pmatrix} \cos(\psi_i)\sin(\theta_i) \\ \sin(\psi_i)\sin(\theta_i) \\ \cos(\theta_i) \end{pmatrix}. \tag{6.7}$$

Abbildung 6.2 stellt diesen Zusammenhang grafisch dar. Häufig wird von dem Fall ausgegangen, dass

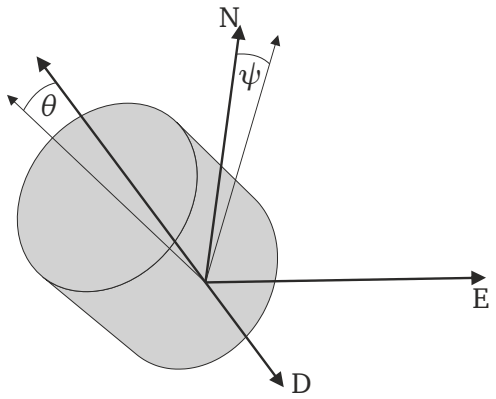


Abbildung 6.2.: Modell für die beliebige Ausrichtung der Motoren.

keine Verdrehung gegenüber dem Body-Koordinatensystem vorliegt, also:

$$\mathbf{f}_i^b = \begin{pmatrix} 0 \\ 0 \\ -f_i \end{pmatrix}. \tag{6.8}$$

Das vereinfacht die Darstellung, lässt aber z. B. keine Modellierung eines sogenannten Pushermotors zu (also ein am Heck ausgerichteter Motor, der ohne Neigung des Multikopters einen Vorwärtsschub erzeugt). Ebenfalls kann durch eine leichte Verkippung der Motoren ein besseres Gierverhalten (also die Drehung um die vertikale Achse) erreicht werden. Ist beispielsweise ein Motor um 5° um den Winkel θ gekippt und um 90° um ψ verdreht, ergibt sich nach Gleichung 6.7 folgender Schubvektor:

$$\mathbf{f}_i^b = -f_i \cdot \begin{pmatrix} 0{,}000 \\ 0{,}087 \\ 0{,}996 \end{pmatrix}. \tag{6.9}$$

Bei einem Verlust von weniger als einem Prozent entlang der vertikalen Achse kann ein Schub in der horizontalen Achse von mehr als 8 % erreicht werden. Diese Kraft kann bei entsprechendem Hebelarm genutzt werden um ein zusätzliches Drehmoment entlang der Gierachse zu erzeugen. Dabei ist allerdings die Drehrichtung des Motors zu beachten, damit sich die Drehmomente nicht aufheben. Mit dieser Verkippung kann zusätzlich ein hohes Drehmoment für das Gesamtsystem erzeugt werden. Gerade für große und schwere bemannte Fluggeräte ist dies von Bedeutung. Siehe beispielsweise Abbildung 6.7: Das Drehmoment ist hier vergleichsweise gering gegenüber dem erzeugten Schub.

Der Gesamtschub der Motoren entspricht im Navigationsframe (n-frame):

$$\mathbf{f}^n = \mathbf{R}_b^n \cdot \sum_{i=1}^{N} \mathbf{f}_i^b. \tag{6.10}$$

Mit der Gleichung für das Drehmoment 6.1 sowie der Gleichung für den Schub 6.10 sind die Bewegungsgleichungen gegeben, um die rotatorische und translatorische Änderung über die Zeit zu beschreiben. Die Objektbeschleunigung $\ddot{\mathbf{x}}^n$ ist abhängig von dem erzeugten Schub der Aktorik, der Abflugmasse m, der Schwerebeschleunigung sowie weiteren einwirkenden Kräften $\mathbf{f}_{\text{Ext.}}^n$:

$$\ddot{\mathbf{x}}^n = \mathbf{g}^n + \frac{1}{m} \cdot \mathbf{f}^n + \frac{1}{m} \cdot \mathbf{f}_{\text{Ext.}}^n. \tag{6.11}$$

Eine wichtige externe Kraft ist in der Luftfahrt selbstverständlich die Strömungswiderstandskraft. Diese Kraft $\mathbf{f}^n_{\text{Air}}$ lässt sich näherungsweise beschreiben:

$$\mathbf{v}_{\text{Airspeed}} = \dot{\mathbf{x}}^n - \mathbf{v}^n_{\text{Wind}} \tag{6.12}$$

$$\mathbf{f}^n_{\text{Air}} = -\frac{1}{2} \cdot A \cdot c_w \cdot \rho \cdot \mathbf{v}_{\text{Airspeed}} \cdot |\mathbf{v}_{\text{Airspeed}}| \tag{6.13}$$

mit der Bezugsfläche A und dem c_w-Wert gegenüber der anströmenden Luft, der Geschwindigkeit $\mathbf{v}_{\text{Airspeed}}$ relativ zur Luft sowie der Luftdichte ρ. Weitere externe Kräfte aber auch Momente können vielfältige Ursachen haben, z. B.:

- Aerodynamik, Kräfte u. Momente werden induziert, abhängig vom Anströmwinkel und der Geschwindigkeit relativ zur Luft
- Schubvariationen, abhängig von der Luftdichte
- Strukturelastizität
- Propellersteifigkeit

Zudem gibt es durch die Anströmung der Propeller weitere Momente (siehe Abbildung 6.3). Strömungsmechanische Effekte sollen hier jedoch nicht vertieft behandelt werden.

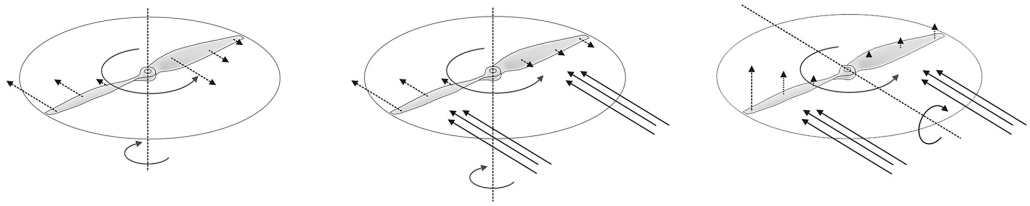
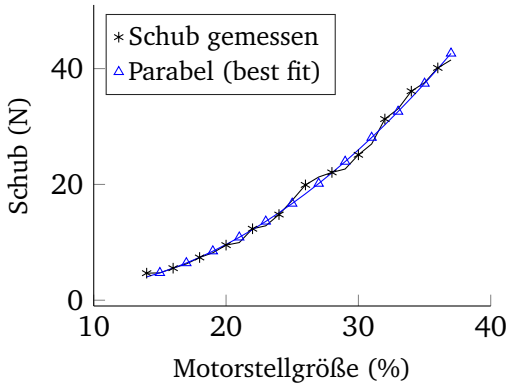
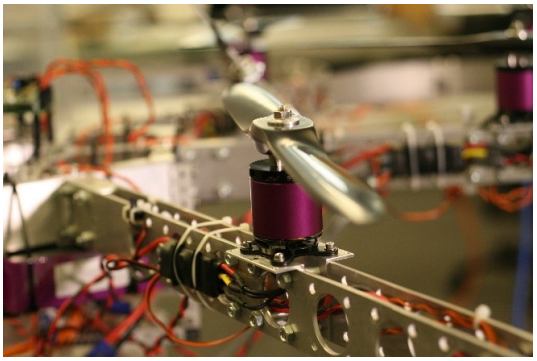


Abbildung 6.3.: Zusätzliche Momente und Kräfte (Propellerdrehung, horizontale Kräfte, Rollmomente) durch Propelleranströmung.

Wenn der maximale Schub für einen Rotor bekannt ist, so ist damit ein 4D Raum definiert, in dem ein Flugregler sich bewegen kann: der skalare Gesamtschub, sowie die Drehmomente der drei Achsen. Abbildung 6.5 zeigt einen 3D Ausschnitt, dabei wird zur Darstellung auf die Gierachse verzichtet. Der mögliche Gesamtschub ist entlang der Hochachse eingezeichnet, die möglichen Drehmomente entlang der Roll- und Pitchachse jeweils auf den horizontalen Achsen. Ein Regelungsverfahren kann sich nun nur in diesem eingegrenzten Raum bewegen. Wenn beispielsweise der maximale Schub abgerufen wird, befindet man sich in der oberen Spitze und hat keine Möglichkeit mehr um Drehmomente zu erzeugen. Bewegt man sich vom Gesamtschub her in der Mitte, so kann man die maximalen Drehmomente erzeugen. Diese Darstellung ist in mehrfacher Hinsicht wertvoll. Man sieht für die Gesamtauslegung, dass das Fluggerät im Schwebeflug näherungsweise in der Mitte des Volumens liegen sollte, zumindest wenn man ein möglichst agiles Fluggerät bauen möchte. Es zeigt sich auch, dass ein Flugregler, der die Höhe regelt, nicht den ganzen Schubbereich ausschöpfen sollte, da sonst keine Reserven übrigbleiben,



(a) Schubkurve.



(b) Installation auf VC25A (siehe auch Abb. 1.5b).

Abbildung 6.4.: Vermessung von Motor *Fa. Hacker, Modell A40-14S V2*.

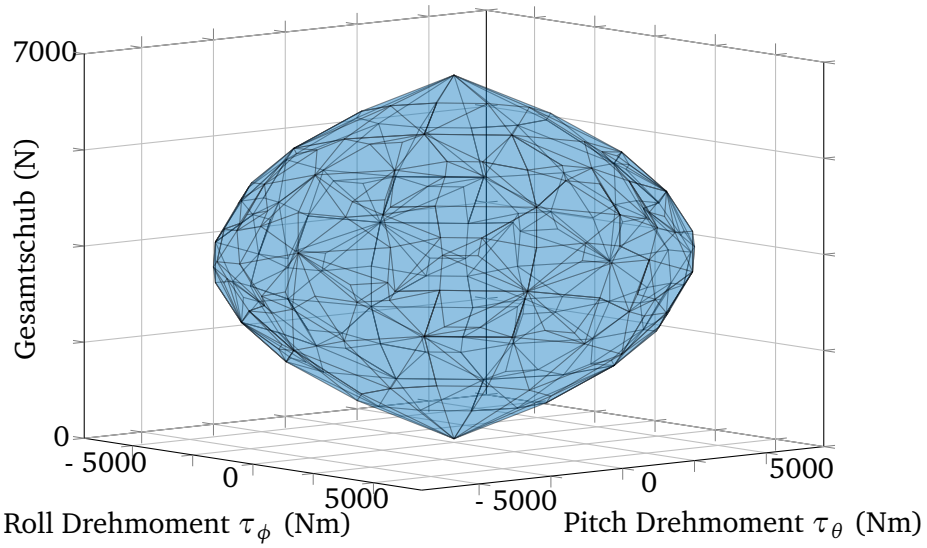


Abbildung 6.5.: Erzeugbare Drehmomente und Kräfte für ein manntragendes Multirotor-Fluggerät, angelehnt an die Darstellungsart von Frangenberg et al. (2015).

um kurzfristige Störungen auszugleichen. Fallen Motoren aus, so verkleinert sich das Volumen. Es weist auch die physikalischen Grenzen auf.

Bisher wurde immer der zu einem Zeitpunkt aufgebrachte Schub der Aktorik betrachtet. Für eine stabile Flugregelung muss aber auch das Antwortverhalten der Antriebseinheit betrachtet werden. Angesteuert werden die (üblicherweise bürstenlosen) Elektromotoren durch die Übertragung der gewünschten Motorstellgröße ($\mathbf{u}$) an die Motorregler (engl. *Electronic Speed Controller*, häufig abgekürzt als ESCs). Motorregler gehören zu den wichtigsten Subsystemen in einem Multirotor-Fluggerät (Puls, 2011). Das Verhalten der Antriebseinheit ergibt sich aus dem Zusammenspiel von Motorregler, Motor, Propeller sowie auch die an den Motorregler anliegende Spannung und der max. abrufbare Strom. Nach Meister (2010) lässt sich das dynamische Übertragungsverhalten gut mit einem PT1-Glied und Totzeit modellieren. Abbildung 6.6 zeigt die Sprungantwort eines Motors, der für ein bemanntes Fluggerät geeignet ist. Vor der Sprungantwort erzeugt das System einen Grundschub von etwa 25 N in einer etwas höheren Leerlaufdrehzahl, da das Antwortverhalten während der Anschleppperiode für die Regelung

nicht von Interesse ist. Je nach Anwendungsfall kann diese Annäherung nicht ausreichend sein, für die allgemeine Systemidentifikation sei auf Ljung (1999) oder Unbehauen (2011) verwiesen. Auch das

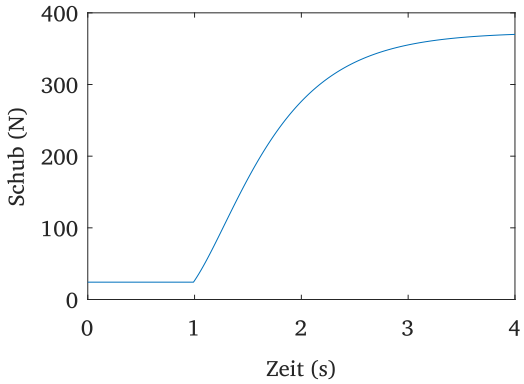


Abbildung 6.6.: Sprungantwort einer Aktorik, modelliert als PT1-Glied, $\tau = 0,6\,s$ (Hacker Q150 Motor).

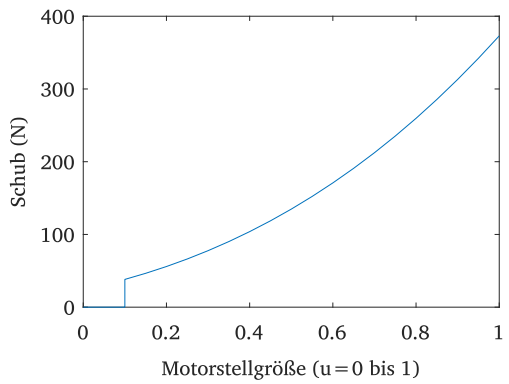
Abbremsverhalten der Aktorik muss auf diese Art modelliert werden. In erster Näherung kann oft das umgekehrte Hochlaufverhalten gewählt werden, aber z. B. bei aktiver Rekuperation kann das Verhalten signifikant unterschiedlich sein. Abbildung 6.7 zeigt ein beispielhaftes Motorverhalten. Zu beachten ist, dass die Modellierung erst ab der Leerlaufdrehzahl beginnt, was in diesem Beispiel bei $u = 0.1$ liegt. Im Flug darf die Motorstellgröße diese Untergrenze dann auch nicht unterschreiten. Bei dem Curve Fitting von Schubkennlinien ist darauf zu achten, dass Messpunkte in den oberen Bereichen ein höheres Gewicht bekommen. Es ist beispielsweise nicht notwendig, dass die Schubkurve im unteren Bereich perfekt das Verhalten des Rotors modelliert, da der erzeugte Schub bzw. die erzeugten Drehmomente nahe der Leerlaufdrehzahl zweitrangig sind. *Beispiel:* Abbildung 6.4a zeigt einen Motor *Fa. Hacker, Modell A40-14S V2*, dessen Schub auf einem Prüfstand gemessen wurde. Die Schubantwort ist hier nichtlinear und lässt sich für einen Motor i durch eine ausgleichende Parabel gut beschreiben[2] (abhängig von der Drehzahl ω_{br} aber auch von der Stellgröße u):

$$\mathbf{f}_i^b = c_1 \cdot \omega_{br}^2 = c_2 \cdot u^2. \tag{6.14}$$

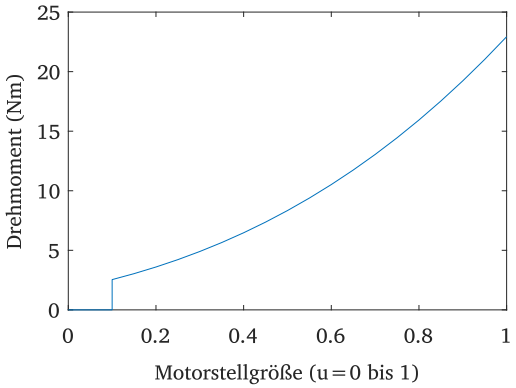
Es ist zu empfehlen, den jeweiligen Koeffizienten c_1 oder c_2 auf einem Prüfstand zu ermitteln.

Der erzeugte Schub eines Propellers ist abhängig von der Luftdichte ρ (siehe Formel 1.1). Somit sind die vorhergehenden Messungen immer auch im Zusammenhang mit den Umweltbedingungen während der Motorvermessung zu sehen. Der dominierende Einfluss auf die Luftdichte ist die Höhe über dem Grund, siehe Abbildung 6.8. In großen Höhen nimmt die Luftdichte offensichtlich ab und damit auch der erzeugbare Schub - bei gleichbleibendem Leistungsbedarf. Dem kann natürlich durch eine höhere Drehzahl (was einem höheren Leistungsbedarf entspricht) entgegengewirkt werden, aber hier werden dann mechanische Grenzen erreicht, z. B. durch Maximalströme oder Maximaldrehzahlen. Neben der Flughöhe spielt noch die Temperatur der Luft, die den Propeller anströmt, eine signifikante Rolle. Desto höher die Temperatur, desto geringer die Luftdichte. Kalte Temperaturen sind daher tendenziell günstig für ein Multirotor-Fluggerät. Die Luftfeuchtigkeit hat in diesem Kontext einen geringen Einfluss auf ρ bzw. auf

[2] Der dargestellte Motor wurde auf dem 25 kg schweren VC25A Test UAV betrieben.



(a) Schub über u.



(b) Drehmoment über u.

Abbildung 6.7.: Antwortverhalten der Antriebseinheit in Abhängigkeit der Motorstellgröße u. Der Schubverlauf beim Anschleppen ist nicht eingezeichnet (Hacker Q150 Motor).

den erzeugbaren Schub und ist vernachlässigbar. Tabelle 6.1 gibt eine Übersicht über die Variationen der

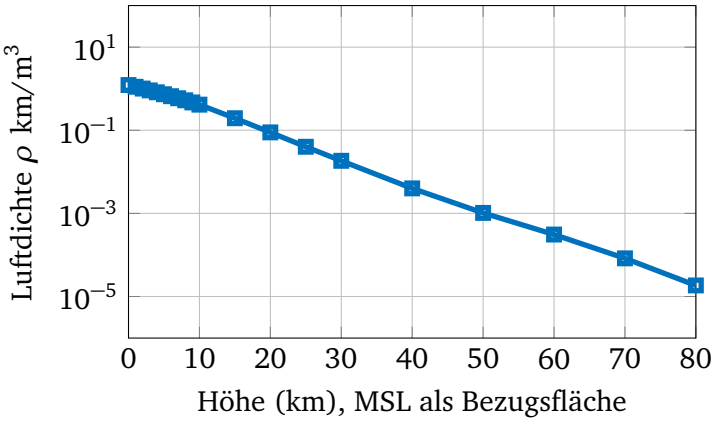


Abbildung 6.8.: Luftdichte ρ in Abhängigkeit der Höhe. Daten von NOAA (1976).

Parameter und den theoretisch erreichbaren Schub. Dabei wird der Leistungsbedarf P bei exakt 60 kW festgehalten und die Umweltparameter variieren. Die Grundlage dafür ist Formel 1.1, umgestellt nach dem Schub T. Diese Parameter wurden gewählt, da sie dem in Abb. 1.5d gezeigten Fluggerät Volocopter VC200 entsprechen. Der Schubunterschied ist deutlich aber nicht dramatisch und kann als Modellabweichung wie z. B. ein Motorausfall, Variation in der Propellereffizienz oder Batteriealterung eingeordnet werden. Denkbar wäre aber auch eine explizite Modellierung. Dies wäre z. B. bei großen Höhen (siehe Abbildung 6.8) empfehlenswert.

Mit einer guten Modellierung der Systemeigenschaften liegt die notwendige Grundlage vor, um a) ein Multirotor-Fluggerät zu simulieren (Abschnitt 6.3) sowie b) um einen Flugregler dafür zu entwerfen und zu testen (Abschnitt 6.4).

6.3 Simulation

Die in Abschnitt 6.2 formelmäßig beschriebenen Zusammenhängen können für eine Simulationsumgebung genutzt werden. Dieser Schritt ist sogar unverzichtbar für bemannte Fluggeräte, da es aus

Tabelle 6.1.: Schub in Abhängigkeit von Temperatur und Flughöhe (Hacker Q150 Motor).

P (W)	**Strahlfläche** (A in m²)	**Höhe** (m)	**Temperatur** (°C)	**Schub** (N)
60000	45	160	0	261,9
60000	45	160	35	251,1
60000	45	1600	0	246,2
60000	45	1600	35	237,4

Sicherheits- und Kostengründen fast nicht möglich ist, mit einer iterativen u. experimentellen Herangehensweise die Flugregelung zu entwerfen. Im Rahmen dieser Arbeit wurde die Simulationsumgebung *multicoptersim* in MATLAB entwickelt, mit dem speziellen Fokus eine Grundlage für die Forschung und Entwicklung mit bemannten Multirotor-Fluggeräten zu schaffen. Abbildung 6.9 zeigt ein simuliertes 18-motoriges Fluggerät in einem lokalen Koordinatensystem. Abbildung 6.10 zeigt die Darstellung von Telemetriedaten aus der Simulationsumgebung; hier kann z. B. der Verlauf von Soll- und Istgrößen nachverfolgt werden. Das Fluggerät wird durch Punktmassen approximiert (Gl. 6.6). Falls ein Massenträgheitsmoment schon vorliegt, beispielsweise aus einem CAD Modell, kann dies auch für die Simulation genutzt werden. In der Simulationsumgebung können beliebige Flugregler als Module eingebunden werden, was für den Reglerentwurf und eine frühe Validierung wertvoll ist. Auch Navigationszustandsschätzer, wie in Kapitel 4 beschrieben, können in dieser Umgebung getestet werden und die Navigationszustandsschätzung kann für die Regleralgorithmen zur Verfügung gestellt werden. Sensorfehler werden gemäß Abbschnitt 2.10 und 3.4 modelliert.

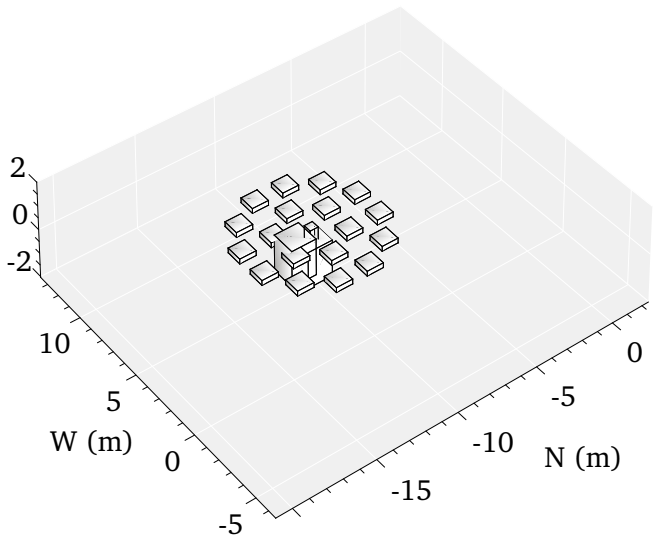


Abbildung 6.9.: Graphische Darstellung innerhalb der Simulationsumgebung (Fluggerät mit 18 Motoren).

6.4 Flugregelung

Auch im Bereich der Regelungstechnik war der Beitrag von Kálmán (1960) signifikant wichtig. Der für die damalige Zeit neue Ansatz in der Regelungstechnik bestand darin den Fokus auf den Zeitbereich und den Zustandsraum zu legen (Müller, 1996). Seit dem Aufkommen von kleinen Multirotor UAVs gibt

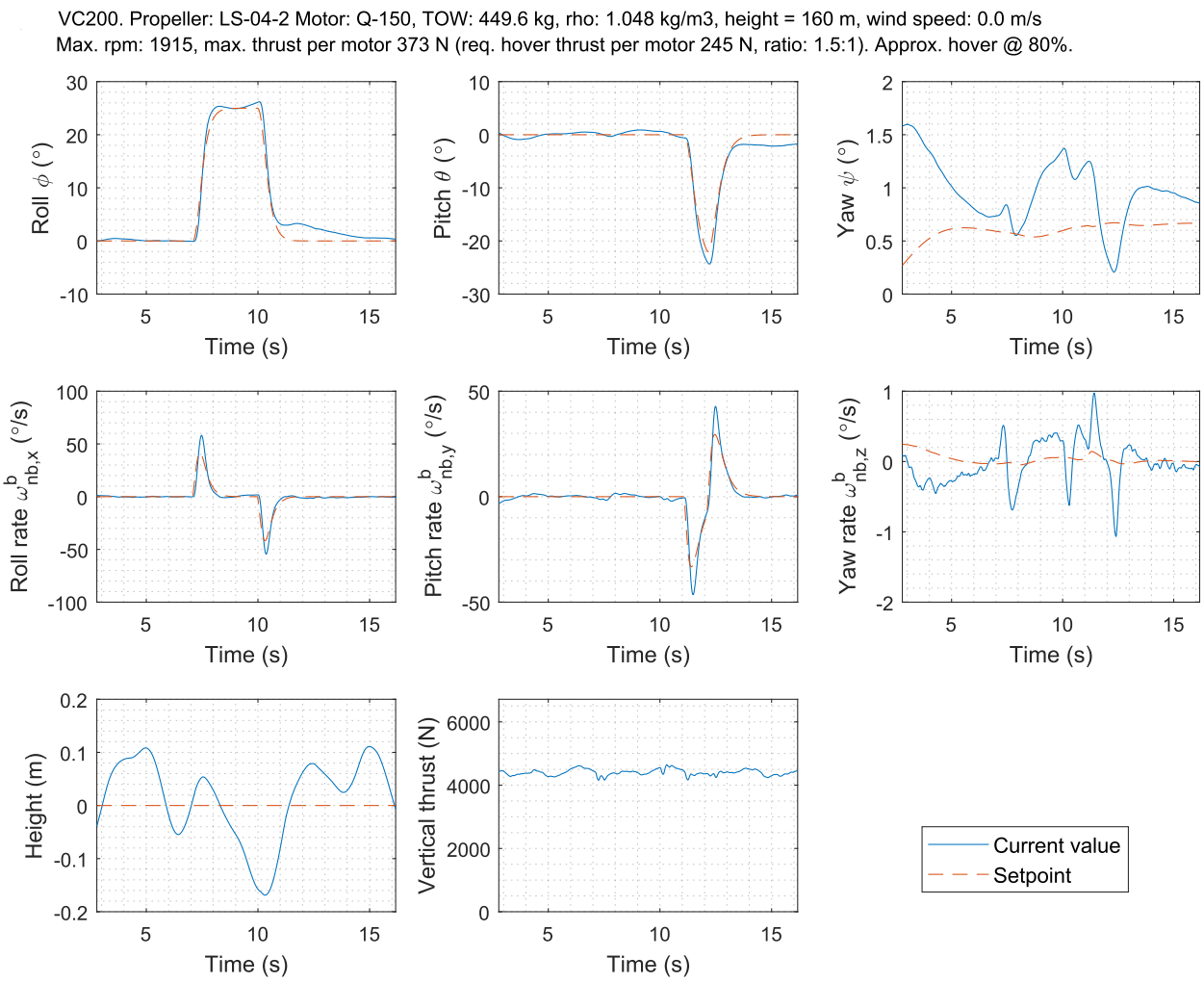


Abbildung 6.10.: Gegenüberstellung von Ist- und Sollwerten (Setpoint Values) in *multicoptersim*.

es eine große Anzahl von Veröffentlichungen, die sich mit deren Lageregelung, Höhenregelung sowie Geschwindigkeits- und Positionsregelung beschäftigen. In einem frühem Beitrag zu dem Thema vergleichen Bouabdallah, Noth und Siegwart (2004) einen Proportional-Integral-Derivative (PID) Regler mit einem linear-quadratischen (LQ) Regler und kommen zu dem Ergebnis, dass der PID-Regler bei kleinen Störungen sowie auch bei dauerhaften Störeinflüssen gut funktioniert, die LQ-Regler Ergebnisse werden aufgrund von Modellfehlern mit *durchschnittlich* bewertet. Meister (2010) vergleicht das Backstepping-Verfahren mit einem PID Regler und zeigt, dass das Backstepping-Verfahren die Sollwerte im Mittel etwas schneller erreicht. Bouabdallah (2007) empfiehlt eine Kombination aus dem PID u. Backstepping Verfahren zur Regelung von Multikoptern. Es vereint die Vorteile des Backstepping (robust gegen Störungen) sowie die Unempfindlichkeit gegenüber Modellabweichungen eines PID Reglers.

Puls (2011) vergleicht verschiedene Lagereglerverfahren (PID, LQ Regler, Feedback Linearization, nichtlineare Regler u. Backstepping) und empfiehlt einen kaskadierten PID-Regler, da dieser robust ist, schnell auf Störungen reagiert, systematische Regelabweichungen ausgleicht und global stabil ist. Argentim et al. (2013) empfehlen einen PID Regler, der durch einen LQ Regler überwacht wird. Häufig werden PID-Regler für Multikopter mit Euler-Winkeln parametrisiert, dies ist aber nicht zwingend.

Der Bereich der adaptiven UAV Flugregelung ist darüber hinaus ein aktives Forschungsthema. Gemeint ist damit eine Flugregelung, die nicht nur tolerant gegenüber Modellfehlern ist (robust), sondern auch versucht die Modellabweichungen zu erlernen. Beispielsweise beschreiben Schumacher und Kumar (2000), wie ein neuronales Netzwerk in einen Fluglageregler integriert werden kann, ohne dass dieses vorab trainiert werden muss. Dydek et al. (2013) zeigen, dass mit einem Model Reference Adaptive Control (MRAC) Verfahren ein Quadcopter bei 25 % Schubverlust noch regelbar bleibt. Bei einem kompletten Schubverlust eines Quadcopter Motors zeigen Mueller und D'Andrea (2014), dass eine kontrollierte Landung noch möglich ist, wenn der Flugregler konstant die Orientierung ändert um so das benötigte Drehmoment auf allen Achsen zu erzeugen.

Su und Cai (2011) beschreiben einen robusten Lageregler, der zwar im Kern auf dem PID Prinzip aufbaut aber die Lage als Quaternion darstellt und auch in der Raumfahrt eingesetzt wird. Su und Cai (2011) zeigen zudem, dass eine kaskadierte Struktur nicht unbedingt notwendig ist. Die Reglerstruktur von Su und Cai (2011) sieht im Kern folgendermaßen aus: Bei einer aktuellen Orientierung des Fluggeräts $\mathbf{q}$ (körperfestes Koordinatensystem b gegenüber Navigationskoordinatensystem n) aus der Navigationszustandsschätzung $\mathbf{y}$ und einer gewünschten Fluglage $\mathbf{q}_d$ kann ein 3×1 Drehmomentvektor $\boldsymbol{\tau}$ zur Lageregelung berechnet werden. Dabei bezeichnet q_0 den skalaren Anteil von $\mathbf{q}_d$ und $\mathbf{q}_v$ den Imaginärteil von $\mathbf{q}_d$. Zuerst wird der Fehler zwischen beiden Lagequaternionen errechnet:

$$\mathbf{B} = \begin{pmatrix} -\mathbf{q}_v^T \\ q_0 \cdot \mathbf{I}_{3\times 3} + [\mathbf{q}_v \times] \end{pmatrix} \tag{6.15}$$

$$\mathbf{e}_v = \mathbf{B}^T \cdot \mathbf{q}_d. \tag{6.16}$$

Dieser Fehlervektor wird bezüglich des Vorzeichens korrigiert und dient dann als Orientierungsvektor für das zu erzeugende Drehmoment:

$$e_0 = \mathbf{q}_d^T \cdot \mathbf{q} \tag{6.17}$$

$$\mathbf{e}_v^\star = \operatorname{sgn}(e_0) \cdot \mathbf{e}_v \tag{6.18}$$

$$\boldsymbol{\tau}^b = \boldsymbol{\tau}_t^b + \mathbf{I}_{e,k} - \left(k_P \cdot \mathbf{e}_v^\star + k_D \cdot \Delta\boldsymbol{\omega}^b\right). \tag{6.19}$$

$\mathbf{I}_{e,k}$ ein Fehlerintegral, $\Delta\boldsymbol{\omega}^b$ ist der aktuelle Drehratenfehler im körperfesten Koordinatensystem, k_P, k_I und k_D die PID Parameter. $\boldsymbol{\tau}_t^b$ ist ein zusätzliches Trimdrehmoment im körperfesten Koordinatensystem, beispielsweise Aufgrund von Center-of-Gravity (CoG) Verschiebungen. Restunsicherheiten werden über das Fehlerintegral behandelt. Das Fehlerintegral wird von Epoche k zu $k+1$ aufintegriert:

$$\mathbf{I}_{e,k+1} = \mathbf{I}_{e,k} + \Delta t \cdot k_I \cdot \mathbf{e}_v^\star. \tag{6.20}$$

Bei Su und Cai (2011) ist die Signumfunktion nicht explizit für die Nullstelle definiert:

$$\operatorname{sgn}(e_0) = \begin{cases} 1, & e_0 \geq 0 \\ -1, & e_0 < 0 \end{cases}. \tag{6.21}$$

Der Drehratenfehler $\Delta\boldsymbol{\omega}^b$ berechnet sich aus der Differenz zwischen der aktuellen Drehrate $\boldsymbol{\omega}_{nb}^b$ und der gewünschten Drehrate $\boldsymbol{\omega}_{nb_d}^b$:

$$\Delta\boldsymbol{\omega}^b = \boldsymbol{\omega}_{nb}^b - \boldsymbol{\omega}_{nb_d}^b. \tag{6.22}$$

Brescianini et al. (2013) beschreiben einen ähnlichen kaskadierten PID Regler, der ebenfalls auf Quaternionen aufbaut und z. B. in der Open Source Flight Control *Pixhawk*[3] eingesetzt wird. Brescianini et al. (2013) betrachten dabei den nicht-trivialen Aspekt, dass ein Fluglageregler die gewünschte Lage zwar allgemein mit der kleinstmöglichen Rotation einnehmen soll, dabei aber der Drehimpuls eine Rolle spielt. Beispielsweise bei Azimutfehlern von 180° kann es schneller sein, den längeren angularen Weg zu nehmen, statt abzubremsen und in der anderen Richtung den etwas kürzeren Weg einzuschlagen. Als Lösung wird ein Schwellwertverfahren vorgeschlagen.

Insgesamt kann man die Lageregelung als den schwierigsten Teil der Multirotor-Flugregelung bezeichnen. Multikopter sind unteraktuierte und dynamisch instabile Systeme (Bouabdallah, Murrieri und Siegwart, 2004) und müssen mit hoher Bandbreite permanent geregelt werden, da sonst innerhalb von Sekunden bzw. Sekundenbruchteilen eine stabile Lage verloren geht. Starrflügler sind im Vergleich dazu eigenstabil.

Höhen- und Positionsregler sind einfacher umzusetzen. Puls (2011) zeigt, dass mit einem PID Höhenregler ein UAV selbst auf einer beweglichen Plattform landen kann. Ein ähnlicher Regler wird in Meister (2010) vorgestellt. Auf das umfangreiche Thema der Trajektorienverfolgung und der Bahnplanung wird hier nicht näher eingegangen. Es sei auf aktuelle Veröffentlichungen wie De Monte (2015) oder Faessler et al. (2018) verwiesen.

Abbildung 6.11 zeigt die Beziehungen der verschiedenen Reglersubsysteme zueinander. Die Führungsgrößen können wieder aus einem übergeordneten Positions- und Geschwindigkeitsregler entstammen oder aus den Steuereingaben des Piloten berechnet werden. Das Ergebnis der Lage- und Höhenregelung ist ein 4×1 Command-Vektor $\boldsymbol{\tau}_{\text{Cmd}}$, bestehend aus dem gewünschten Drehmoment $\boldsymbol{\tau}^b$ in Nm und dem Grundschub f_0 in Newton:

$$\boldsymbol{\tau}_{\text{Cmd}} = \begin{pmatrix} \boldsymbol{\tau}^b \\ f_0 \end{pmatrix}. \tag{6.23}$$

Diese Vorgaben müssen im nächsten Schritt von den Motoren umgesetzt werden.

[3] URL: `http://pixhawk.org/`, Stand 15. Okt. 2018.

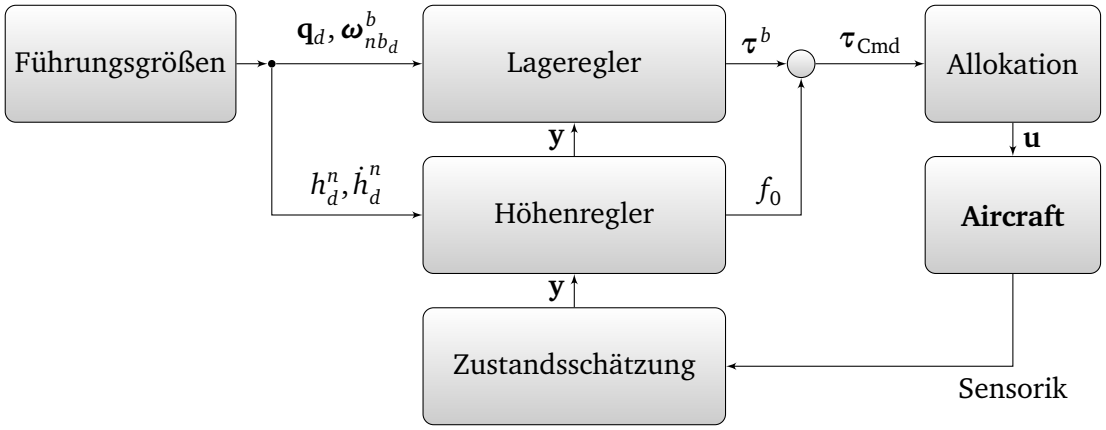
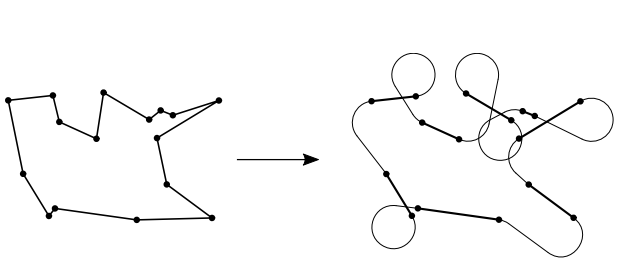
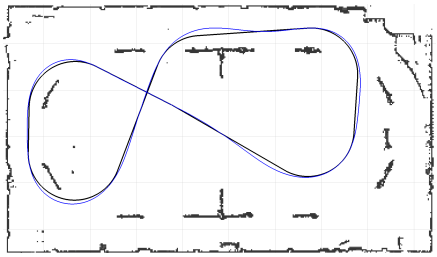


Abbildung 6.11.: Blockschaltbild Lage- und Höhenregelung von Multirotor-Fluggerät.

6.5 Aspekte des autonomen Fliegens



(a) Einbeziehung der Flugdynamik in der Abfolge von Wegpunkten. Abbildung aus Savla et al. (2005).



(b) Beschreibung der Trajektorie mit Polynomen (blau) in komplexer Umgebung gegenüber Dubins Pfaden (schwarz). Abbildung aus Bry et al. (2015).

Abbildung 6.12.: Aspekte der Bahnplanung für autonome Flugmissionen.

Eine grundlegende Motivation für die Betrachtung der robusten Navigationszustandsschätzung in dieser Arbeit ist, dass für autonome oder automatisierte Systeme die Anforderungen an die Zuverlässigkeit signifikant steigen. Das betrifft zudem nicht nur Fluggeräte, sondern beispielsweise auch Fahrzeuge oder Roboter. Während bei einem pilotierten Flug eine falsche Zustandsschätzung durch einen menschlichen Eingriff kompensiert werden kann, gibt es für einen rein maschinellen Flug keine derartige Risikominimierung. Für das autonome Fliegen sind inhärent sichere Konzepte zur Navigationszustandsschätzung notwendig.

Darüber hinaus müssen weitere Aufgaben von Avionik Systemen übernommen werden. Allen voran die Bahnplanung. Da Fluggeräte nicht beliebig schnell ihre Orientierung und Geschwindigkeit ändern können, wird die Bahnplanung häufig auf der Basis von Dubins Pfaden gestützt (Dubins, 1961). Dubins Pfade erlauben die Berechnung der kürzesten Trajektorie zwischen zwei Punkten, unter der Berücksichtigung von Nebenbedingungen. Die Dubins Pfadsegmente lassen sich aus Geraden und Kreisbögen zusammensetzen. Die Krümmung wird von den kinematischen Eigenschaften des Fluggeräts bestimmt, also den erreichbaren Drehraten und Geschwindigkeiten, man spricht auch von einem *Dubins Vehicle* (Hota und Ghose, 2014).

Als Optimierungskriterium kann auch beispielsweise der Treibstoffverbrauch gewählt werden. Werden derartige Eigenschaften in die Pfadplanung einbezogen, können Trajektorien errechnet werden, die a) zur Flugdynamik des Fluggeräts passen und b) treibstoffoptimiert sind (Zhou und Prasad, 2014).

Soll ein Dubins Vehicle verschiedene Orte mit einer möglichst kurzen Wegstrecke erreichen, so ergibt sich ein *Traveling Salesman* Optimierungsproblem. Mögliche Lösungsalgorithmen werden in Savla et al. (2005) betrachtet. Abbildung 6.12a zeigt beispielhaft, wie die flugdynamischen Eigenschaften in die *Traveling Salesman* Lösung einbezogen werden müssen.

Für die Trajektorienplanung in komplexen Umgebungen mit agilen Fluggeräten und aggressiven Manövervorgaben kann es lohnenswert sein, die Bahnplanung auf Polynombasis zu berechnen (Bry et al., 2015). Abbildung 6.12b zeigt eine Bahnplanung mit Polynomen.

Für dynamisch erkannte Hindernisse während der Mission kann der $A*$ Algorithmus genutzt werden, um Wegpunkte für Ausweichmanöver zu berechnen (Selecký et al., 2013).

Mögliche Sensorsystemkonzepte zur Hinderniserkennung für manntragende Multirotor-Fluggeräte werden in Rupalla (2018) betrachtet. Hervorzuheben sind in diesem Zusammenhang folgende Sensoren:

- Kameras
- Infrarotkameras
- Lidar
- Radar
- kooperativen Systeme (beispielsweise ADS-B)

Auch die dynamisch geschätzten Windverhältnisse können in die Pfadplanung einfließen. Solch ein adaptiver Ansatz kann genutzt werden, um die Flugzeiten zu optimieren und um den Komfort zu erhöhen, ohne dass die Regelgesetze geändert werden. Zudem können Sicherheitsabstände zu Hindernissen angepasst werden, siehe Benders et al. (2018). Benders et al. (2018) stellen zudem einen Zustandsschätzer vor, der den Windgeschwindigkeitsvektor ohne direkte Sensoren zur Beobachtung ermittelt.

6.6 $\mathscr{L}_1$ Motorallokation

Um die gewünschten Momente und Kräfte zu erzeugen, muss den individuellen Motoren eine Stellgröße zugewiesen werden. Dies wird Motorallokation bzw. *Control Allocation* (Härkegård und Glad, 2005), Thrust Mixing (Faessler et al., 2017) oder auch Motormixing genannt (Paparazzi, 2015). Quan (2017) beschreibt das Grundproblem als das Lösen eines Gleichungssystems mit Nebenbedingungen. Ein interdisziplinärer Überblick findet sich in Johansen und Fossen (2013), z. B. neben der Luft- und Raumfahrt: maritime Anwendungen, Industrieanlagen oder im Automotive-Bereich. Also wie muss der Motorstellgrößenvektor $\mathbf{u}$ aussehen, damit möglichst $\boldsymbol{\tau}_{\mathrm{Cmd}}$ als Systemantwort erreicht wird? Die Nebenbedingungen geben vor, dass die Stellgrößen zwischen der Leerlaufdrehzahl und der max. Drehzahl liegen. Hinzu kommt, dass mit der Anzahl der Motoren auch die Anzahl der Möglichkeiten wächst, das Problem zu lösen.

Faessler et al. (2017) schlagen einen iterativen Algorithmus für Quadrocopter vor, der in Sättigungssituationen folgende Priorisierung durchführt: Roll- und Pitchmomente bekommen Vorrang vor

dem Gesamtschub und dieser wird wieder vor dem Giermoment priorisiert. Diese Priorisierung ist sinnvoll, ohne eine stabilisierte Lage ist letztendlich auch keine Höhenstabilisierung möglich. In Stephan und Fichter (2017) wird ein iterativer Algorithmus für beliebige Aktorikkonfigurationen vorgestellt. Dieser Algorithmus passt iterativ $\boldsymbol{\tau}_{\text{Cmd}}$ in das vorhandene *Control Volume* ein (siehe Abbildung 6.5), unter der Bedingung, dass die Richtung von $\boldsymbol{\tau}_{\text{Cmd}}$ erhalten bleibt. Andernfalls würden gerade schwierige Flugsituationen, welche die Motoren in die Sättigung treiben, unbeabsichtigte Drehmomente erzeugen.

Für einen Stellgrößenvektor $\boldsymbol{\Omega}$, der aus den quadrierten Motordrehzahlen Ω_i besteht, kann folgender Zusammenhang zu $\boldsymbol{\tau}_{\text{Cmd}}$ hergestellt werden für N Motoren (Frangenberg et al., 2015):

$$\boldsymbol{\tau}_{\text{Cmd}} = \underbrace{\begin{pmatrix} \bar{\mathbf{m}}_1^b & \bar{\mathbf{m}}_2^b & \cdots & \bar{\mathbf{m}}_N^b \\ \bar{f}_1^b & \bar{f}_2^b & \cdots & \bar{f}_N^b \end{pmatrix}}_{\mathbf{K}} \cdot \underbrace{\begin{pmatrix} \Omega_1^2 \\ \Omega_2^2 \\ \vdots \\ \Omega_N^2 \end{pmatrix}}_{\boldsymbol{\Omega}}. \tag{6.24}$$

Die Drehzahlen lassen sich (je nach Aktorikeigenschaften) mit den Motorstellgrößen in Beziehung setzen (siehe Gl. 6.14):

$$\Omega_i^2 = f\left(u_i\right). \tag{6.25}$$

Die Vektoren $\bar{\mathbf{m}}_i^b$ modellieren das erzeugte Drehmoment vom i-ten Motor, wenn mit der quadrierten Motordrehzahl Ω_i^2 multipliziert wird (vgl. Gl. 6.2). Die Drehrichtung und das Drehmomentverhalten der Motoren muss über geeignete Modelle in diesen Vektoren enthalten sein, beispielsweise über Faktoren, die auf einem Propellerprüfstand ermittelt wurden. Ebenso modelliert $\bar{f}_i^b$ den erzeugten Schub vom i-ten Motor, wenn mit der quadrierten Motordrehzahl Ω_i^2 multipliziert wird. Die gesuchte Umkehrung von $\mathbf{K}$ in Gl. 6.24 ist über die Pseudoinverse möglich. Von Ducard und Hua (2011) wird eine gewichtete Pseudoinverse vorgestellt, die sich folgendermaßen berechnet:

$$\mathbf{K}^+ = \mathbf{W}^{-1} \cdot \mathbf{K}^T \cdot \left(\mathbf{K} \cdot \mathbf{W}^{-1} \cdot \mathbf{K}^T\right)^{-1}. \tag{6.26}$$

Für eine allgemeine Betrachtung der Pseudoinversen sei z. B. auf Jäger et al. (2005) verwiesen, numerisch robuster kann zudem eine Matrix Decomposition sein. In Gl. 6.26 muss $\mathbf{K} \cdot \mathbf{K}^T$ invertierbar sein. Wenn das nicht gegeben ist, so ist die Motorplatzierung derartig ungünstig, dass nicht für alle Achsen ein Drehmoment erzeugt werden kann. Die Lösung der Pseudoinversen minimiert die Quadrate der Abstände für dieses überbestimmte Problem, was hier ein gewolltes Kriterium darstellt, da i. A. die Bewegungen mit möglichst kleinen Stellgrößenänderungen erzeugt werden sollen. Die Gewichtsmatrix $\mathbf{W}^{-1}$ ist optional. Sie kann aber genutzt werden, um Motoren zu bevorzugen oder diese weniger zu belasten. Im ersten Schritt kann somit $\boldsymbol{\Omega}$ berechnet werden:

$$\boldsymbol{\Omega} = \mathbf{K}^+ \cdot \boldsymbol{\tau}_{\text{Cmd}}. \tag{6.27}$$

Im ungesättigten Fall, wenn also das untere Drehzahllimit $\underline{\Omega}$ nicht unterschritten und das obere Limit $\overline{\Omega}$ nicht überschritten wird, kann die Lösung $\mathbf{\Omega}$ direkt genutzt werden. Werden die Zwangsbedingungen nicht für alle N Motoren eingehalten ($\forall_i \in N : \underline{\Omega} \leq \mathbf{\Omega}_i \leq \overline{\Omega}$), so muss $\boldsymbol{\tau}_{\text{Cmd}}$ skaliert werden. Dies kann als *Quadratic Programming* Problem verstanden werden; darüber hinaus wird in Oppenheimer et al. (2006) die Allokation als *Linear Programming* Problem behandelt, was direkt mit den Methoden in Kapitel 5 gelöst werden kann, aber dann eben nicht mehr die Abstandsquadrate minimiert. Bodson (2002) zeigt, wie der Simplex $\mathscr{L}_1$ Algorithmus für die *Control Allocation* genutzt werden kann. Ein $\mathscr{L}_2$-Norm Algorithmus (*Redistributed Pseudoinverse*) mit einer oberen Laufzeitschranke wird von Stephan und Fichter (2017) beschrieben. Von Frost und Bodson (2010) wird eine *Control Allocation* mit der $\mathscr{L}_\infty$-Norm vorgestellt. Die $\mathscr{L}_\infty$-Norm minimiert die minimale und maximale Aktorikansteuerung (Mini-Max Optimierung) und kann als Lastverteilung angesehen werden. Die $\mathscr{L}_\infty$-Norm, bzw. Tschebycheff-Anpassung ist ein Sonderfall der M-Schätzer, siehe Abschnitt 2.5 (Jäger et al., 2005). Die $\mathscr{L}_\infty$-Norm lässt sich darüber hinaus für die Control Allocation als Linear Programming (LP) Problem mit Hilfe der $\mathscr{L}_1$-Norm lösen, siehe Bodson und Frost (2011).

Wenn zum Zeitpunkt der Allokation bekannt ist, dass einzelne Motoren nicht funktionsfähig sind, dann kann dies direkt in der Allokation selbst berücksichtigt werden. Der Vorteil ist, dass der Regelalgorithmus dadurch entlastet wird. Die Kommandos in $\boldsymbol{\tau}_{\text{Cmd}}$ haben nach wie vor ihre Gültigkeit, der Schub und die Momente müssen nur von den übrigen Motoren aufgebracht werden. Ein Motorausfall kann entweder direkt durch Rückmeldungen der Motorsteuerlogik erkannt werden, aber auch durch eine Zustandsschätzung. Frangenberg et al. (2015) schlagen dazu einen Least-Squares Schätzer vor, der iterativ für jeden Motor prüft, ob ein hypothetischer Motorausfall das aktuelle Bewegungsverhalten besser erklärt. Dies setzt ein gutes Fluggerätemodell voraus, siehe Abschnitt 6.2.

Im Folgenden soll ein Verfahren für Motorallokation vorgestellt werden, das die Homogenisierung aus Abschnitt 2.6 in die $\mathscr{L}_1$ Motorallokation übernimmt und darüber hinaus sicherstellt, dass die erzeugten Drehmomente in Sättigungssituationen nicht verzerrt werden. Der Ausgangspunkt ist Gleichung 6.24:

$$\boldsymbol{\tau}_{\text{Cmd}} = \mathbf{K} \cdot \begin{pmatrix} \Omega_1^2 \\ \Omega_2^2 \\ \vdots \\ \Omega_N^2 \end{pmatrix} \tag{6.28}$$

$$\boldsymbol{\tau}_{\text{Cmd}} = \mathbf{K} \cdot \mathbf{\Omega}. \tag{6.29}$$

Gesucht wird ein Vektor an Propellerdrehraten $\mathbf{\Omega}$, um die gewünschten Drehmomente und den Gesamtschub in $\boldsymbol{\tau}_{\text{Cmd}}$ zu erzeugen. Für alle Drehraten gilt, wie bereits erläutert, dass die oberen und unteren Limits nicht überschritten werden dürfen:

$$\forall i \in \{1 \ldots N\} : \underline{\Omega}_i^2 \leq \Omega_i^2 \leq \overline{\Omega}_i^2. \tag{6.30}$$

Dabei ist es zulässig für jeden Motor individuelle obere Schranken zu setzen. Die Ungleichungen für die $\mathscr{L}_1$ Motorallokation lauten somit:

$$\begin{pmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & & \\ \vdots & & \ddots & \vdots \\ 0 & & \dots & 1 \end{pmatrix} \cdot \begin{pmatrix} \Omega_1^2 \\ \Omega_2^2 \\ \vdots \\ \Omega_N^2 \end{pmatrix} \leq \begin{pmatrix} \overline{\Omega}_1^2 \\ \overline{\Omega}_2^2 \\ \vdots \\ \overline{\Omega}_N^2 \end{pmatrix} \tag{6.31}$$

$$-\begin{pmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & & \\ \vdots & & \ddots & \vdots \\ 0 & & \dots & 1 \end{pmatrix} \cdot \begin{pmatrix} \Omega_1^2 \\ \Omega_2^2 \\ \vdots \\ \Omega_N^2 \end{pmatrix} \leq \begin{pmatrix} -\underline{\Omega}_1^2 \\ -\underline{\Omega}_2^2 \\ \vdots \\ -\underline{\Omega}_N^2 \end{pmatrix}. \tag{6.32}$$

Bzw. in kurzer Form:

$$\mathbf{I}_{N\times N} \cdot \boldsymbol{\Omega} \leq \overline{\boldsymbol{\Omega}} \tag{6.33}$$

$$-\mathbf{I}_{N\times N} \cdot \boldsymbol{\Omega} \leq -\underline{\boldsymbol{\Omega}}, \tag{6.34}$$

zusammengefasst zu

$$\mathbf{E} = \left(\frac{\mathbf{I}_{N\times N}}{-\mathbf{I}_{N\times N}}\right) \tag{6.35}$$

$$\mathbf{f} = \left(\frac{\overline{\boldsymbol{\Omega}}}{-\underline{\boldsymbol{\Omega}}}\right). \tag{6.36}$$

Gleichung 6.29 sowie die Ungleichungen 6.33 und 6.34 stellen damit die Grundlage für die $\mathscr{L}_1$ Motorallokation dar. Diese Form ist aber noch nicht ausreichend. In Sättungungssituationen, also wenn mehr Kräfte und Drehmomente gefordert werden als die Motoren liefern können, kommt es zur Verzerrung von $\boldsymbol{\tau}_{\text{Cmd}}$. Abbildung 6.13a zeigt beispielhaft, wie das angeforderte Drehmoment für die Roll- und Pitchachse nach der Motorallokation abweichen kann. Die Folge ist eine ungewollte Lageänderung. Um dies zu verhindern, wird eine neue Bedingungsgleichung eingeführt. Ein Vektor $\mathbf{d}$ wird berechnet, der orthogonal zu $\boldsymbol{\tau}^b$ (Gl. 6.23) steht:

$$\mathbf{d}^T \cdot \boldsymbol{\tau}^b = 0. \tag{6.37}$$

Die Bedingung ist nun, dass der aus der Allokation resultierende Drehmomentvektor $\boldsymbol{\tau}^b_{\text{allok.}}$ ebenfalls orthogonal zu $\mathbf{d}$ stehen muss. Oder anders ausgedrückt: $\boldsymbol{\tau}^b$ darf nur skaliert werden. Die Bedingungsgleichung lautet damit:

$$\begin{pmatrix}\mathbf{d}^T & 0\end{pmatrix} \cdot \mathbf{K} \cdot \boldsymbol{\Omega} = 0. \tag{6.38}$$

Danach gilt für das resultierende Drehmoment der $\mathscr{L}_1$ Motorallokation:

$$\mathbf{d}^T \cdot \boldsymbol{\tau}^b_{\text{allok.}} = 0. \tag{6.39}$$

Abbildung 6.13b zeigt den Einfluss von Gl. 6.38. Der gewünschte Drehmomentvektor ändert seine Richtung nicht und nutzt das Kontrollvolumen optimal aus. Der Vektor $\mathbf{d}$ kann beispielsweise über ein Kreuzprodukt berechnet werden:

$$\mathbf{d} = \boldsymbol{\tau}^b \times \begin{pmatrix}1 & 0 & 0\\ 0 & 0 & -1\\ 0 & 1 & 0\end{pmatrix} \cdot \boldsymbol{\tau}^b. \tag{6.40}$$

Ein weiterer wichtiger Bestandteil des vorgestellten Verfahrens ist die Homogenisierung der Motormatrix $\mathbf{K}$ sowie von $\boldsymbol{\tau}_{\text{Cmd}}$ aus Gl. 6.29. Wie in Abschnitt 2.6 beschrieben, werden $\mathbf{K}$ und $\boldsymbol{\tau}_{\text{Cmd}}$ über eine Matrix $\mathbf{T}$ homogenisiert:

$$\bar{\boldsymbol{\tau}}_{\text{Cmd}} = \mathbf{T} \cdot \boldsymbol{\tau}_{\text{Cmd}} \tag{6.41}$$

$$\bar{\mathbf{K}} = \mathbf{T} \cdot \mathbf{K} \tag{6.42}$$

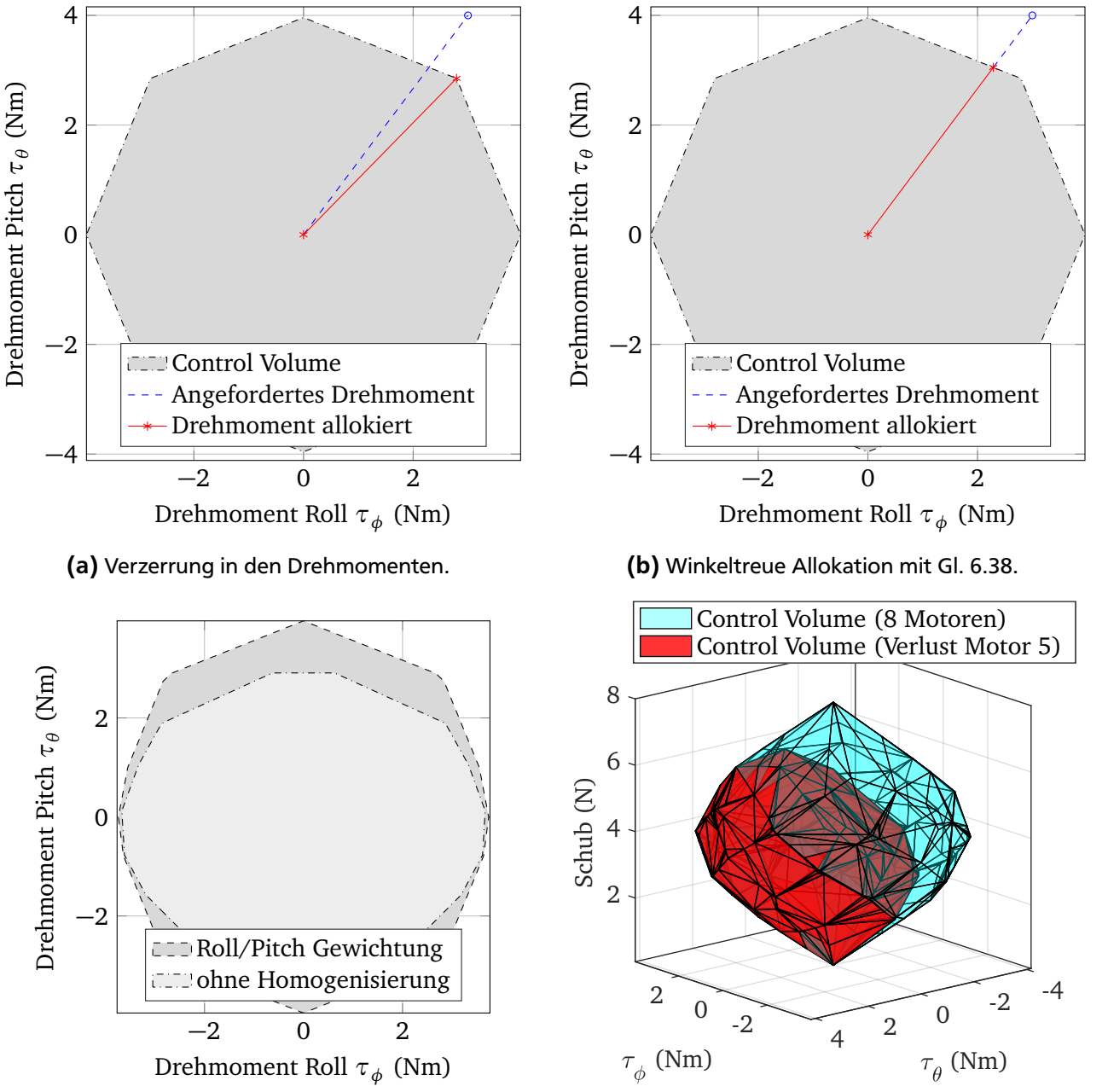
mit den Gewichten p_ϕ für den Rollwinkel, p_θ für den Pitchwinkel, p_ψ für den Gierwinkel und p_f für den Gesamtschub:

$$\mathbf{T} = \begin{pmatrix}\sqrt{p_\phi} & 0 & 0 & 0\\ 0 & \sqrt{p_\theta} & 0 & 0\\ 0 & 0 & \sqrt{p_\psi} & 0\\ 0 & 0 & 0 & \sqrt{p_f}\end{pmatrix}. \tag{6.43}$$

Für Multikopteranwendungen ist es sinnvoll, wenn beispielsweise die Roll- und Pitch-Drehmomente höher gewichtet werden als der Grundschub und die Gierachse. Ohne stabile Lagewinkel kann auch die Höhe nicht sicher geregelt werden. Auch ist eine verzögerte Regelung der Gierachse eher hinzunehmen als ein Verlust der Gesamtlage. Abbildung 6.13c zeigt eine Priorisierung der Roll- und Pitch-Drehmomente um den Faktor 10. Ohne diese Gewichtung bzw. Homogenisierung ist das Kontrollvolumen deutlich kleiner. Diese Ausweitung geht dafür zu Lasten des Gesamtschubs bzw. der Gierachse.

Das vorgestellte Verfahren lässt sich mit dem Algorithmus aus Anhang B einfach umsetzen (siehe Gl. 6.41, 6.42, 6.38, 6.35 u. 6.36):

$$\mathbf{\Omega} = \text{CL1NORM}\left(\bar{\mathbf{K}}, \bar{\boldsymbol{\tau}}_{\text{Cmd}}, \begin{pmatrix}\mathbf{d}^T & 0\end{pmatrix} \cdot \bar{\mathbf{K}}, 0, \mathbf{E}, \mathbf{f}\right). \tag{6.44}$$



(a) Verzerrung in den Drehmomenten.

(b) Winkeltreue Allokation mit Gl. 6.38.

(c) Priorisierung der Roll- u. Pitch-Achse.

(d) Einschränkung des *Control* Volumens nach einem Motorausfall bei einem UAV mit 8 Motoren.

Abbildung 6.13.: Motorallokation mit der $\mathscr{L}_1$ Methode.

Die Kontrollvolumina in Abb. 6.13 wurden auf Basis der Motormatrix eines Octocopters[4] berechnet:

$$\mathbf{K} = \begin{pmatrix} 0.59 & 1.41 & 1.41 & 0.59 & -0.59 & -1.41 & -1.41 & -0.59 \\ 1.44 & 0.56 & -0.56 & -1.44 & -1.44 & -0.56 & 0.56 & 1.44 \\ -1.00 & 1.00 & 1.00 & -1.00 & -1.00 & 1.00 & 1.00 & -1.00 \\ 1.00 & 1.00 & 1.00 & 1.00 & 1.00 & 1.00 & 1.00 & 1.00 \end{pmatrix}. \tag{6.45}$$

Beispiel: Gegeben sei ein Steuerbefehlsvektor:

$$\tau_{\text{Cmd}} = \begin{pmatrix} 3.00 \\ 4.00 \\ 0.00 \\ 2.00 \end{pmatrix} \tag{6.46}$$

sowie eine Gewichtsmatrix **T**:

$$\mathbf{T} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0.1 & 0 \\ 0 & 0 & 0 & 0.1 \end{pmatrix}. \tag{6.47}$$

Wird dieser Vektor inklusive der Motormatrix **K** über Gl. 6.43 homogenisiert, ergeben sich folgende Werte:

$$\bar{\mathbf{K}} = \begin{pmatrix} 0.59 & 1.41 & 1.41 & 0.59 & -0.59 & -1.41 & -1.41 & -0.59 \\ 1.44 & 0.56 & -0.56 & -1.44 & -1.44 & -0.56 & 0.56 & 1.44 \\ -0.10 & 0.10 & 0.10 & -0.10 & -0.10 & 0.10 & 0.10 & -0.10 \\ 0.10 & 0.10 & 0.10 & 0.10 & 0.10 & 0.10 & 0.10 & 0.10 \end{pmatrix} \tag{6.48}$$

$$\bar{\tau}_{\text{Cmd}} = \begin{pmatrix} 3.00 \\ 4.00 \\ 0.00 \\ 0.20 \end{pmatrix}. \tag{6.49}$$

[4] Die Matrix **K** des Octocopters wurde von Frangenberg et al. (2015) übernommen.

Fällt für diesen Octocopter ein Motor aus, beispielsweise Motor Nr. 5, so ergibt sich eine neue Motormatrix $\mathbf{K}$:

$$\bar{\mathbf{K}} = \begin{pmatrix} 0.59 & 1.41 & 1.41 & 0.59 & \mathbf{0.00} & -1.41 & -1.41 & -0.59 \\ 1.44 & 0.56 & -0.56 & -1.44 & \mathbf{0.00} & -0.56 & 0.56 & 1.44 \\ -0.10 & 0.10 & 0.10 & -0.10 & \mathbf{0.00} & 0.10 & 0.10 & -0.10 \\ 0.10 & 0.10 & 0.10 & 0.10 & \mathbf{0.00} & 0.10 & 0.10 & 0.10 \end{pmatrix}. \tag{6.50}$$

Das Kontrollvolumen reduziert sich dadurch signifikant, dies ist für das genannte Beispiel in Abbildung 6.13d zu sehen.

Das vorgestellte Verfahren kann als robuste Motorallokation für ein Multirotor-Fluggerät genutzt werden. Gerade die Priorisierung der Lage gegenüber der Höhenregelung ist sinnvoll. Besonders Fluggeräte mit einem ungünstigen Schub/Gewichtsverhältnis profitieren davon.

6.7 $\mathscr{L}_1$-basierte robuste Auswahllogik für die Aktorik

Der $\mathscr{L}_1$ Simplex Algorithmus mit Ungleichungen kann dazu beitragen, die Ausfallsicherheit von einem Fluggerät mit redundanter Reglerkonfiguration zu erhöhen. Das Verfahren aus diesem Abschnitt wurde bereits in Jäger und Zwiener (2016) veröffentlicht.

Ein Fluggerät, insbesondere ein bemanntes, darf keinen Single Point of Failure enthalten. Das macht redundante Konfigurationen notwendig. Das Thema wird seit Jahrzehnten behandelt, besonders für Starrflügler, siehe z. B. Stengel (1973) u. Ahlstrom et al. (2002). Bei Passagierflugzeugen ist es *State of the Art*, dass die Flugregelung mehrfach redundant und dissimilar[5] umgesetzt wird um gemeinsame Fehler (engl. *Common Mode Failure*) zu vermeiden (Yeh, 1996). Multirotor UAV Flugsteuerungen haben in den meisten Fällen keine Architektur, die frei ist von Single Point of Failures (SPoF) und praktisch nie mit dissimilarer Soft- u. Hardware. Im Folgenden wird eine neue Redundanzarchitektur für Multirotor-Fluggeräte gezeigt, die keine SPoFs enthält und auch einfach um Dissimilarität erweitert werden kann.

Abbildung 6.14 zeigt den grundsätzlichen Aufbau. Es sind k Sensoren an m Flugsteuerungen (Flight Controls) angeschlossen, diese wiederum senden für n Motorsteuerungen einen Vektor mit Motorstellgrößen $\mathbf{u}$ (siehe Abschnitt 6.6). Jede der m Flugsteuerungen berechnet aus allen verteilten k Sensoren für sich einen Navigationszustandsvektor $\mathbf{y}_j$ mit $j \in \{1, \ldots, m\}$. Denkbar ist aber auch, dass die Anzahl der Sensorverbände k genau der Anzahl der Flugsteuerungen m entspricht ($k = m$) und jede der Flugsteuerungen genau einem Sensorarray zugeordnet ist. Dieser Ansatz würde zwar die Genauigkeit der Zustandsschätzung herabsetzen, es wäre aber sichergestellt, dass einzelne Sensorfehler nicht zu einem *Common Mode* Problem für alle Zustandsschätzer führen. Bancroft (2010) beschreibt dieses Szenario für Multi-IMU Systeme in der Luftfahrt und argumentiert, dass hier die Fehlererkennung (und Isolierung) wichtiger ist als ein Genauigkeitsgewinn.

Jeder Motor i mit $i \in \{1, \ldots, n\}$ muss für sich entscheiden, welcher der m Motorstellgrößenvektoren $\mathbf{u}_1, \mathbf{u}_2, \ldots \mathbf{u}_m$ als Closed-Loop Lösung gewählt wird. Die Vektoren $\mathbf{u}$ haben die Dimension $n \times 1$ und so

[5] Entwickelt von räumlich und organisatorisch getrennten Teams, die unterschiedliche Algorithmen verwenden, mit unterschiedlicher Hardware arbeiten und mit unterschiedlicher Software sowie Compilern entwickeln.

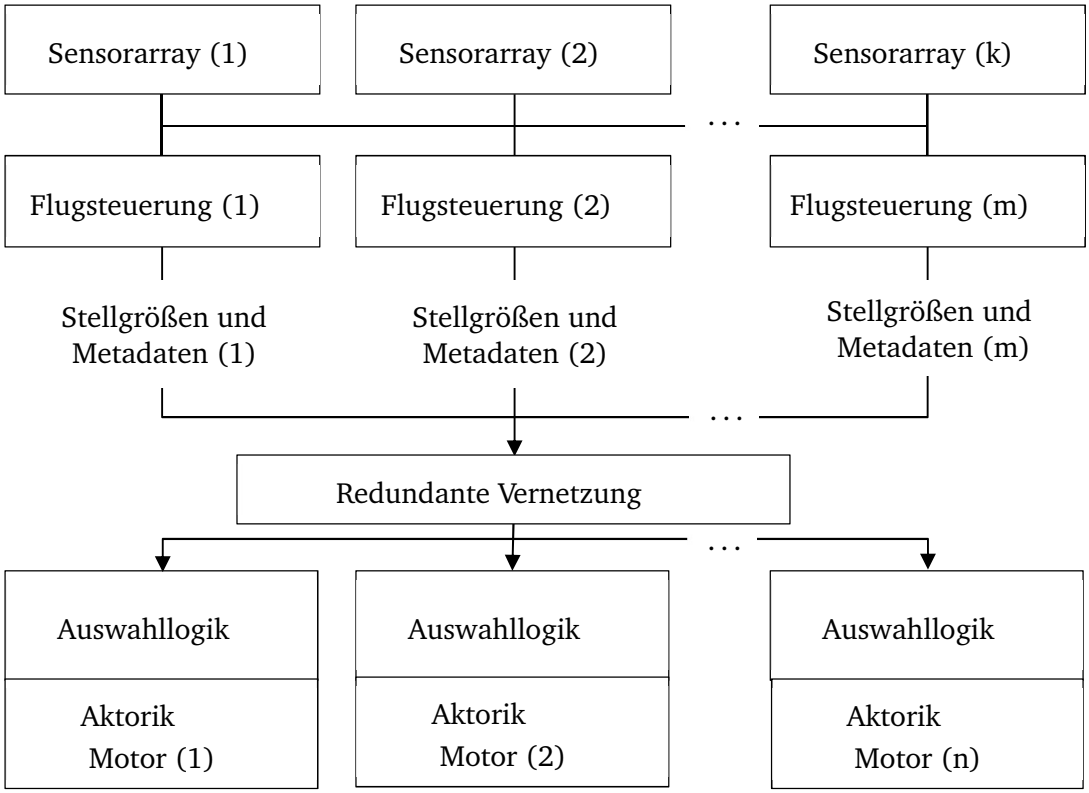


Abbildung 6.14.: Aufbau für eine redundante Flugregelung.

nimmt jeder Motor für sich die i-te Motorstellgröße aus dem gewählten Vektor $\mathbf{u}$. Dabei wird angenommen, dass die Motorstellgrößen $\mathbf{u}_i$ so verteilt werden, dass alle Motoren in demselben Zeitfenster Δt die gleichen Motorstellgrößen zur Verfügung haben. Fällt eine Flugsteuerung komplett aus, so ist für die jeweilige Flugsteuerung kein Motorstellgrößen in dem jeweiligen Zeitfenster verfügbar. Alle in dem Zeitfenster verfügbaren Motorstellgrößen $\mathbf{u}_j$ bildet jeder Motor i zu dem Vektor

$$\mathbf{l}_i = \left(\mathbf{u}_1^T, \mathbf{u}_2^T, \ldots, \mathbf{u}_m^T\right)^T. \tag{6.51}$$

Somit kann über das Simplex Verfahren auf jedem Motor i ein Voting durchgeführt werden, um den gewählten Stellgrößenvektor $\mathbf{u}_v$ zu bestimmen:

$$\mathbf{l}_i = \mathbf{A}_i \cdot \mathbf{u}_v \tag{6.52}$$

mit

$$\begin{pmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \\ \vdots \\ \mathbf{u}_m \end{pmatrix}_i = \begin{pmatrix} \mathbf{I}_{n\times n} \\ \mathbf{I}_{n\times n} \\ \vdots \\ \mathbf{I}_{n\times n} \end{pmatrix}_i \cdot \mathbf{u}_v. \tag{6.53}$$

Nun könnte Gleichung 6.53 auch nach der Methode der kleinsten Quadrate ($\mathscr{L}_2$ Norm) gelöst werden, dann wäre jedoch der Einfluss von Fehlern (durch eine fehlerhafte Berechnung in einer Flugsteuerung) nicht begrenzt, siehe Einflussfunktion $\psi(\bar{v})$ für die $\mathscr{L}_2$ Norm in Gl. 2.60. Dies ist bei der $\mathscr{L}_1$ Norm gegeben. Denkbar wären theoretisch auch andere robuste Schätzer. Der Vorteil der $\mathscr{L}_1$ Norm ist abgesehen von der Robustheit, dass sich eine Lösung mit den Methoden der linearen Programmierung bzw. der linearen Optimierung auf Basis von modifizierten Simplex-Algorithmen finden lässt. Dadurch eröffnet sich die Möglichkeit auch Ungleichungen in folgender Form einzubeziehen:

$$\mathbf{E} \cdot \mathbf{u}_v \leq \mathbf{f}. \tag{6.54}$$

Die Matrix $\mathbf{E}$ hat konstant die Dimension $2n \times n$, unabhängig von der Anzahl der verfügbaren Motorstellgrößen. Auch der Vektor $\mathbf{f}$ bleibt konstant bei der Dimension $2n \times 1$. Typische Werte für den Inhalt von $\mathbf{f}$ wären $\overline{u} = 1$ (für 100 %) sowie eine untere Schranke von den Stellgrößen von $\underline{u} = 0$ (0 %). Damit wird sichergestellt, dass sich die Motorstellgrößen alle in einem gültigen Bereich bewegen $\forall_i \in n : \underline{u} \leq u_i \leq \overline{u}$.

$$\begin{pmatrix} \mathbf{I}_{n\times n} \\ \hline -\mathbf{I}_{n\times n} \end{pmatrix} \cdot \mathbf{u}_v \leq \left(\begin{array}{c} \overline{u}_1 \\ \overline{u}_2 \\ \vdots \\ \overline{u}_n \\ \hline -\underline{u}_1 \\ -\underline{u}_2 \\ \vdots \\ -\underline{u}_n \end{array}\right). \tag{6.55}$$

Um zu vermeiden, dass sich die kommandierten Momente durch die Ausgleichung des Vektors $\mathbf{u}_v$ ergeben, wird im Anschluss an das Voting der Vektor $\mathbf{u}_j$ gewählt, der die niedrigste Abstandsbetragssumme zu dem gewählten Vektor $\mathbf{u}_v$ hat. Damit sind Unstetigkeiten verbunden, die aber in der Praxis kaum erkennbar sind. Eine Alternative dazu ist, tatsächlich direkt mit dem Ergebnisvektor $\mathbf{u}_v$ zu fliegen. Aufgrund der $\mathscr{L}_1$ Eigenschaften sind die Übergänge bei einem Wechsel der Flugsteuerung in $\mathbf{u}_v$ stetig, solange die Stellgrößen selbst näherungsweise stetig sind.

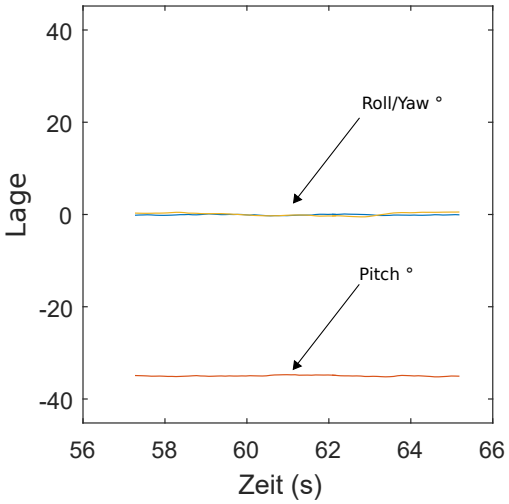
Ein signifikanter Aspekt ist, dass immer nur eine Flugsteuerung pro Motor aktiv ist, also sich im Closed-Loop Modus befindet. Die übrigen $m-1$ Flugsteuerungen sind im Open-Loop Modus. Das kann z. B. einen negativen Einfluss auf die Integratoren der Open-Loop Flugregler haben, die sich dann evtl. in eine Sättigung bewegen. Es gibt verschiedene Ansätze damit umzugehen. Eine Möglichkeit ist es, die Motorstellgrößen in einen deterministischen Teil d und in einen adaptiven Teil a zu trennen:

$$\mathbf{u}_j = \mathbf{u}_j^d + \mathbf{u}_j^a. \tag{6.56}$$

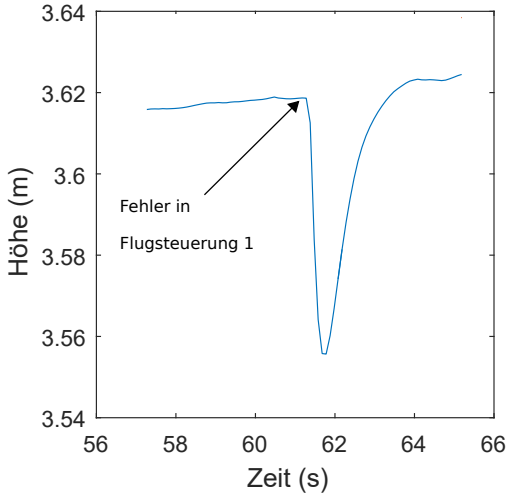
Die letztendlich kommandierte Stellgröße ist in $\mathbf{u}_j$ enthalten, das beschriebene Voting findet aber mit dem deterministischen Anteil $\mathbf{u}_j^d$ statt (bei einem PID Regler beispielsweise der Stellgrößenvektor ohne

I-Anteil). Die Flugsteuerungen führen selbst das Voting durch und erkennen so, ob sie sich im Open-Loop Modus befinden. Wenn sich eine Flugsteuerung im Open-Loop Modus befindet, so kann sie den adaptiven Teil in $\mathbf{u}_j^a$ festhalten, bis wieder ein Übergang in den Closed-Loop-Modus stattfindet.

Experimentelle Prüfung:
Ein Multirotor-UAV wurde im Geradeausflug (in Richtung der Roll-Achse) mit einem Pitch-Winkel von -35° in der Simulationsumgebung *multicoptersim* (siehe Abschnitt 6.3) geflogen. Dabei wurde in Sekunde 61,137 die aktive Flugsteuerungslösung $j = 1$ mit Zufallszahlen überschrieben, um einen groben Fehler in $\mathbf{u}_1$ zu simulieren (Abb. 6.15). Die Flugsteuerungen j=2 und j=3 (Gesamtanzahl an Flugsteuerungen: $m = 3$) liefen weiter. In der Lage ist das Umschalten nicht zu erkennen (Abb. 6.15a). Die Auswirkungen sind dafür zumindest im Höhenregler (Abb. 6.15b) zu erkennen, hier verliert das UAV kurzzeitig zwischen Sek. 61 und 64 etwa 7 cm an Höhe. Das ist auf den adaptiven Teil im Höhenregler der neu-aktivierten Flugsteuerung (hier j=2) zurückzuführen. Wäre für die Lagestabilisierung ein höherer I-Anteil notwendig, so wäre hier auch ein transientes Verhalten zu beobachten.



(a) Lage (Roll/Yaw bei 0°, Pitch bei −35°).



(b) Tatsächlicher Höhenverlauf.

Abbildung 6.15.: Simulation von groben Fehlern in Motorstellgrößenvektor $\mathbf{u}_j$ mit $j = 1$. Ausfall bei Sekunde 61.

Wahlverhalten unter normalen Bedingungen:
Bei gleichen Sensoren und näherungsweise gleicher Einbauorientierung der Sensoren zeigt sich bei dem vorgestellten Konzept, dass die Flugsteuerungen von den dezentralisierten Antriebseinheiten gleichverteilt gewählt werden. Abbildung 6.16 zeigt ein Histogramm von einem simulierten Multikopter Flug. Die Wahrscheinlichkeit für eine Flugsteuerung gewählt zu werden, liegt bei etwa $p = 0.33$ bzw. 33 %, siehe folgende Tabelle:

FC 1	FC 2	FC 3
34.5 %	32.0 %	33.5 %

Bei verschiedenen Durchläufen gibt es Variationen, entsprechend sind in Abbildung 6.16 Fehlerbalken von sechs verschiedenen Durchläufen eingezeichnet. Diese Variationen haben ihre Ursache in der Simu-

lation des Sensorrauschens sowie einem simulierten Random-Walk in den Sensor Offsets. Das zeitliche

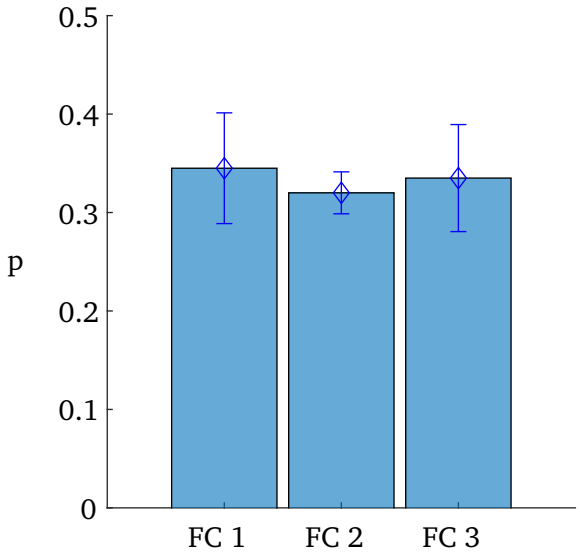


Abbildung 6.16.: Wählverhalten bei drei gleichen Flugsteuerungen.

Verhalten des Median-Votings weißt ebenfalls eine Gleichverteilung auf. Abbildung 6.17 zeigt die gewählte Flugsteuerung (1, 2 oder 3) über die Zeit.

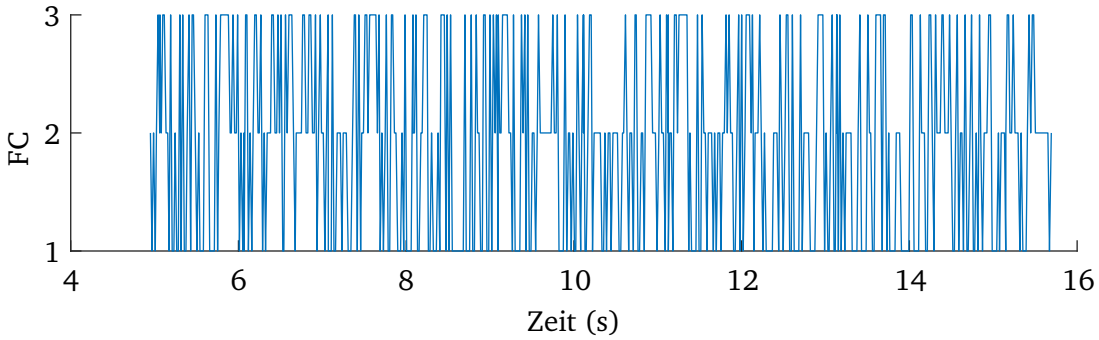


Abbildung 6.17.: Zeitliches Verhalten der $\mathscr{L}_1$ Voting Architektur.

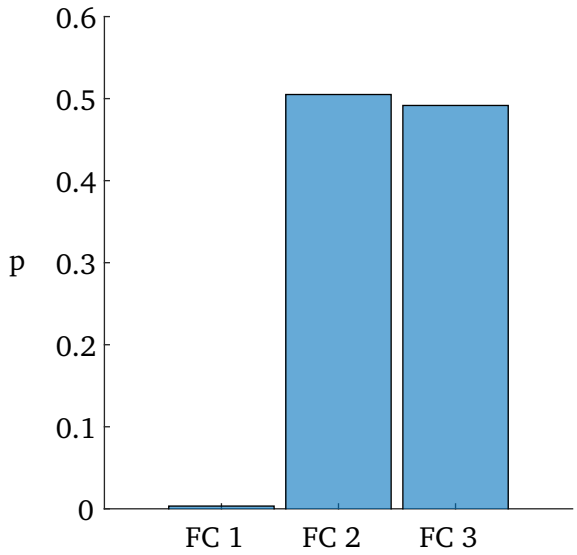
Wahlverhalten bei groben Fehlern:
Werden von einer Flugsteuerung falsche (oder auch gar keine) Steuervektoren **u** ausgegeben, so kann man direkt an der Verteilung abschätzen, welche Flugsteuerung betroffen ist. In Abbildung 6.18a sieht man die Verteilung einer Simulation, bei der von Flugsteuerung 1 ein dauerhaft grob fehlerhafter Vektor **u** berechnet wurde. Entsprechend sind die Wahrscheinlichkeiten zu etwa 50 % zwischen Flugsteuerung 2 und 3 aufgeteilt.

Wahlverhalten bei Sensorrauschen:
Ist die Navigationszustandsschätzung einer Flugsteuerung durch hohes Sensorrauschen beeinflusst, so hat dies ebenfalls sichtbare Auswirkungen auf die Wahlverteilung. In einer Simulation wurde das Gyro-

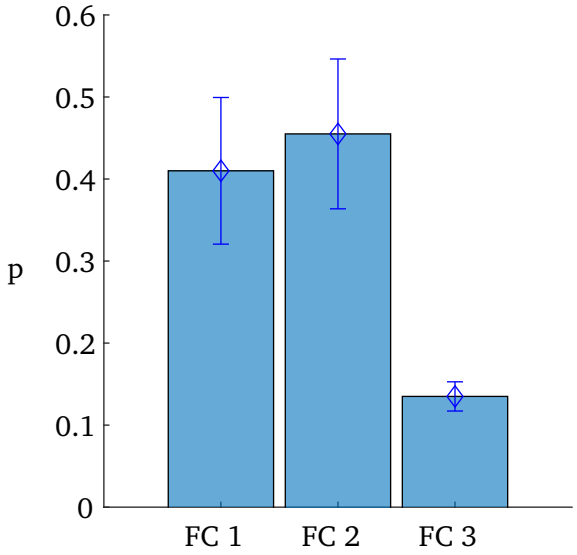
skoprauschen einer Flugsteuerung vergleichsweise hoch eingestellt. Folge Tabelle zeigt das (normalverteilte) zeitdiskrete Rauschen der simulierten Gyroskope ($v \sim \mathcal{N}(0, \sigma_\omega)$):

	FC 1	**FC 2**	**FC 3**
σ_ω	$0.91°/s$	$0.91°/s$	$2.75°/s$

Abbildung 6.18b zeigt die Verteilung der gewählten Flugsteuerung. Im Gegensatz zu einem groben Fehler wird hier Flugsteuerung 3 durchaus gewählt, aber deutlich seltener als die anderen Flugsteuerungen mit geringerem Sensorrauschen.



(a) Grob fehlerhafte Daten von Flugsteuerung 1.



(b) Erhöhtes Sensorrauschen in Flugsteuerung 3.

Abbildung 6.18.: Wählverhalten bei Abweichungen zwischen den Flugsteuerungen.

6.8 Störeinflüsse auf barometrischen Messungen während dem Start und der Landung

Bei Start- und Landemanövern von Fluggeräten können Barometer signifikant gestört werden, insbesondere bei Kampfjets (Gray und Maybeck, 1995). Das gilt auch für Multikopter.

Desto größer die Gesamtpropellerfläche des Multikopters ist, desto mehr Luft strömt nach unten ab. Dadurch erhöht sich der Luftdruck p (in Pascal) in Bodennähe unterhalb der Rotorebene. Die barometrische Höhenformel von Tanigawa et al. (2008) abgeleitet nach dem Luftdruck lautet (siehe Abschnitt 3.6):

$$h(p) = 44300 \cdot \left(1 - \left(\frac{p}{p_0}\right)^{0,19}\right) \tag{6.57}$$

$$\frac{\partial h}{\partial p} = -\frac{8417}{p_0^{0,19}} \cdot p^{-0,81} \tag{6.58}$$

$$= -942,044 \cdot p^{-0,81}. \tag{6.59}$$

An Gleichung 6.59 sieht man direkt am negativen Vorzeichen, dass eine Luftdruckzunahme zu einer Reduzierung der barometrischen Höhe h führt. Bei Multikoptern führt das zu einer Beeinträchtigung der Navigationslösung, falls ein Barometer in das Navigationssystem integriert ist. In Abbildung 6.19 sieht man die Auswirkungen bei dem 25 kg schweren VC25a Multikopter. Ab Sekunde 40 hebt der Multikopter ab und schwebt dann mit etwa 1 m Abstand zum Boden, bis er bei etwa Sekunde 140 wieder landet. Die Motoren werden ab Sekunde 148 deaktiviert. Die Kreise markieren die kritischen Phasen. Während dem Start und der Landung erhöht sich der Luftdruck in Bodennähe, sodass die Navigationslösung eine zu niedrige Höhe berechnet. Der Effekt ist bei größeren (400-500 kg) Multikoptern stärker. Ungefähr

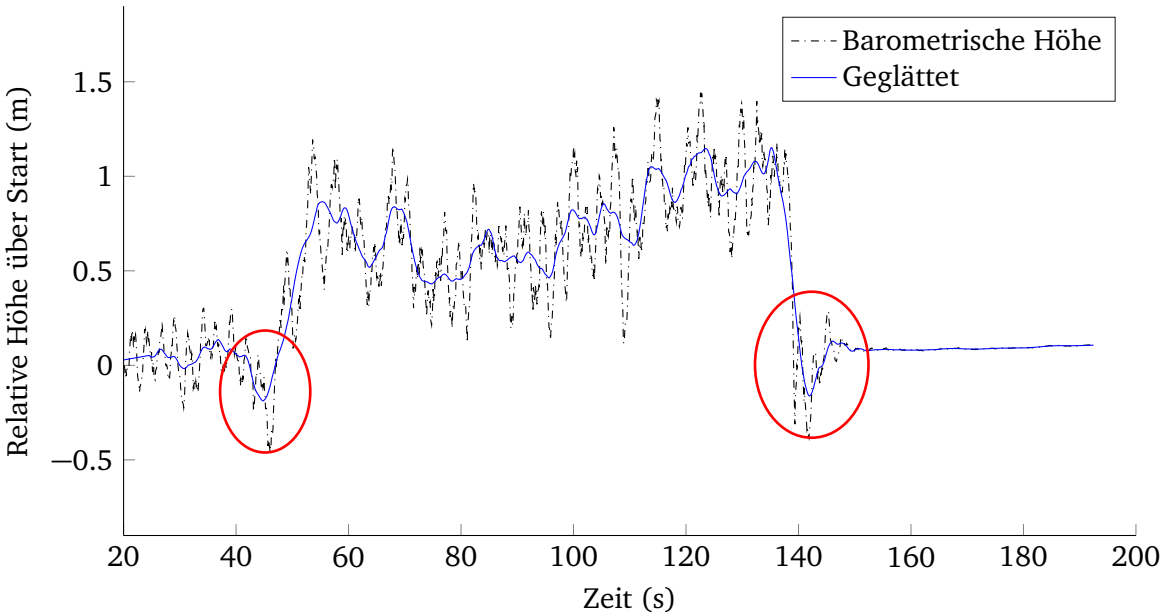


Abbildung 6.19.: Einfluss von Aktorik (Motor/Propeller) auf die Navigationslösung. Markiert ist die Start- und Landungephase.

5 Meter Abweichung sind hier beim Start beobachtbar. Entsprechend gefährlich können die Auswirkungen auf die Zustandsschätzung und Höhenregelung sein. Mögliche Ansätze zur Korrektur sind:

- Robuste Parameterschätzung (siehe Abschnitt 5.1)
- Heruntergewichten der Barometermessungen in Bodennähe
- Zusätzliche Sensoren für den Start und die Landung
- Korrekturen der Barometermessungen abhängig von der Motordrehzahl in Bodennähe
- Barometerplatzierung auf Luftfahrzeug optimieren

Wenn möglich kann das Problem durch ein Versetzen des Barometers umgangen werden. Bei großer räumlicher Trennung sollte dann allerdings der Hebelarm zwischen Barometer und IMU berücksichtigt werden, siehe Abschnitt 3.6. Mit vergleichsweise wenig Aufwand kann das Problem durch dynamisches Heruntergewichten der Barometermessungen in Bodennähe umgangen werden. Der Einfluss auf die Barometermessung in Bodennähe kann durch eine Kalibrierung korrigiert werden. Z. B. in Form eines Korrekturpolynoms, welches zur Barometermessung aufaddiert wird. Diese Lösung erfordert jedoch eine plattformspezifische Einmessung.

In Tests hat sich folgendes Verfahren bewährt: ein robuster Zustandsschätzer verwirft Barometermessungen, die nicht zur Verteilung der bisherigen Zustandsschätzung passen. Dies löst auf einfache Weise das beschriebene Problem. Dies an sich kann jedoch nicht ausreichend sein. Befindet sich das Fluggerät im Modus einer nicht-redundanten und reduzierten Zustandsschätzung (siehe Abschnitt 4.5), so kann dies zu falschen Schätzungen führen, da beim Start auch IMU Messungen fehlerhaft sein können. Ohne GNSS Stützung ist es möglich, dass die angebrachte Korrektur der Accelerometerbiase ungenau ist. Schon kleine Biasfehler führen durch die notwendige Doppelintegration zu signifikanten Höhenfehlern. Durch ein zu optimistisches stochastisches Modell der IMU kann es nun passieren, dass der reinen Inertialnavigation zu stark vertraut wird und die Barometermessungen übermäßig verworfen werden. Als Verfahren wird Folgendes vorgeschlagen: Die IMU *coasting* Zeit wird für den reduzierten Zustandsschätzer (Lage u. Höhe) auf wenige Sekunden beschränkt. Sollte also der robuste Zustandsschätzer über diese Zeit hinaus alle Barometermessungen verwerfen, so ist es zu empfehlen, die Barometermessungen für eine bestimmte Zeit (*cooldown-period*) nicht zu verwerfen. Diese *cooldown-period* sollte etwa das zehnfache der *coasting-period* sein. Eben aus dem Grund, dass das stochastische Modell der Zustandsschätzung selbst grob fehlerhaft sein kann. In diesem reduzierten Szenario steht es sozusagen „Aussage gegen Aussage“ zwischen IMU und Barometer. Aber auch für redundante Konfigurationen mit vielen Sensoren ist dieses Verfahren empfehlenswert. Beispielsweise für den Sonderfall, dass die Kalibrierung aller IMUs grob fehlerhaft ist.

7 Zusammenfassung

Die Untersuchung von GNSS/MEMS gestützten Navigationsverfahren für Multirotor-Fluggeräte ist Gegenstand dieser Arbeit. Basierend auf der Grundidee, die Effizienz eines Schätzers gegenüber Robustheit und Bruchpunktresistenz zu tauschen, wurde das Simplex Kalman-Filter vorgestellt. Die Möglichkeit, direkt Ungleichungen über das Simplex Verfahren einzubeziehen, wurde genutzt um mit der Methode der binären Raumteilung (BSP) beliebige konkave Raumgeometrien als Beschränkung einzuführen. Durch die Einführung von zusätzlichem Wissen, z. B. aus Kartenmaterial, kann die absolute Genauigkeit der Zustandsschätzung erhöht werden. Gegenüber einem Partikelfilter ist der Rechenaufwand hier geringer, sodass diese Methode auch auf Computersystemen mit geringer Rechenleistung oder Arbeitsspeicher genutzt werden kann.

Natürlich sind robuste Kalman-Filter Verfahren durchaus üblich und auch Ungleichungen oder Bedingungsgleichungen lassen sich in ein $\mathscr{L}_2$ Kalman-Filter integrieren. Denkbar wäre es auch, die vorgestellte BSP Methode in ein $\mathscr{L}_2$ Kalman-Filter zu integrieren. Dieser Ansatz wurde im Rahmen dieser Arbeit nicht verfolgt, wenngleich dies unter Umständen betrachtenswert ist. Der Vorteil bei der $\mathscr{L}_1$ Methode ist die hohe Bruchpunktsresistenz. Beispielsweise konnten in den vorgestellten Auswertungen 1/3 der GNSS Rohdaten grob fehlerhaft sein.

Aufgrund dessen, dass Ungleichungen und Bedingungen inhärent in den Simplex Algorithmus integriert sind, ist die Anwendung und Handhabung in der Praxis vergleichsweise einfach. Diese Eigenschaften bekommt man, im Gegensatz zur $\mathscr{L}_2$ Norm, sozusagen „*for free*“. Es sei auch auf die robuste Umsetzung der CL1NORM Funktion hingewiesen, die ganz allgemein ein wertvolles Werkzeug ist, das nicht nur mit beliebig groben Fehlern umgehen kann, sondern mit Floating Point Darstellungen wie *INF* oder *NAN* ohne Mehraufwand umgehen kann. Das Simplex Verfahren eröffnet also neue Möglichkeiten, ist sehr robust und benötigt im Betrieb wenig Aufmerksamkeit.

Diese Vorteile erkauft man sich mit der geringeren Genauigkeit. So wurde mit synthetischen normalverteilten Pseudoranges gezeigt, dass die $\mathscr{L}_2$ Norm in Kombination mit einer IMU auf eine Genauigkeit (für den beschriebenen Aufbau) von 0,3 m ($1\,\sigma$) kommt, während die $\mathscr{L}_1$ Norm mit den gleichen Messungen nur 2 m Genauigkeit ($1\,\sigma$) erreicht. Natürlich kann die $\mathscr{L}_2$ Norm bei normalverteilten Daten ohne Ausreißer nicht übertroffen werden, aber es ist für jeden Anwendungsfall zu prüfen, ob die Genauigkeitsverluste hinnehmbar sind.

Die Fehlerfortpflanzung bzw. die Berechnung der Kovarianzmatrix nach dem Simplex Kalman-Filter Korrekturschritt hat sich numerisch als schwieriger als erwartet gezeigt. Dies kann über das vorgestellte QR-Zerlegungsverfahren gelöst werden. Insgesamt ist die $\mathscr{L}_1$ Norm Schätzung mit Ungleichungen keine lineare Funktion, sodass hier die Darstellung der Verteilung über eine Kovarianzmatrix an Grenzen stößt.

Eine Besonderheit des Simplex Kalman-Filters ist, dass für die hohe Bruchpunktresistenz die Beobachtungen gleichzeitig verarbeitet werden müssen. Dafür eignen sich GNSS Messungen hervorragend. Für Messungen, die zeitlich verteilt eintreffen, ist diese Eigenschaft unter Umständen limitierend.

Es wurde ein Redundanzkonzept für Multirotor-Fluggeräte vorgestellt, das auf dem Simplex $\mathscr{L}_1$ Verfahren aufbaut. Es kann gezeigt werden, dass damit in einer Anordnung von mindestens drei Flugsteuerungscomputern jede Art von Fehler in einem der Computer kompensiert werden kann. Durch die Verteilung der $\mathscr{L}_1$ Auswahllogik auf die Aktorik entsteht auch kein Single-Point-of-Failure.

Darüber hinaus wurden weitere Aspekte betrachtet:

Die vorgestellten $\mathscr{L}_2$ Kalman-Filter Gleichungen in der Takasu Form sind, soweit bekannt, bisher nicht in der Literatur zu finden, aber aufgrund der numerischen Stabilität erwähnenswert.

Eine komplette Übersicht über die GNSS/MEMS Gleichungen unter Berücksichtigung der Hebelarmeffekte wurde gezeigt und kann als Formelsammlung für Navigationsanwendungen mit verteilten Sensoren verstanden werden.

Es wurde untersucht, ob die Verlet Integrationsmethode für die Integration von IMU Messungen geeignet ist. Es kann aber keine Empfehlung ausgesprochen werden, da dieses Verfahren geringfügig schlechtere Ergebnisse liefert als die Standardverfahren (1,091 m Fehler gegenüber 1,005 m Fehler der Trapezregel).

Was in dieser Arbeit nicht bearbeitet wurde, ist die Möglichkeit der Einführung von Ganzzahlbedingungen in das Simplex Kalman-Filter. Hier ergeben sich interessante neue Möglichkeiten. Jedoch zählt die ganzzahlige lineare Optimierung im Sinne der Komplexitätstheorie zur Klasse der *NP-schweren* Probleme. Das macht dieses Verfahren ab einer gewissen Problemgröße unattraktiv, aber möglicherweise wäre die Einbeziehung der Lösung von ganzzahligen Mehrdeutigkeiten bei der relativen Positionierung mit GNSS Phasenmessungen handhabbar.

A Quellcode Simplex Kalman-Filter

Der folgende Quellcode implementiert in der MATLAB Programmiersprache das Simplex Kalman-Filter Konzept (z. B. verwendet in Abschnitt 5.4).

```
function [dx, Qxx] = simplex_kalman(l, H, x, Qxx, Qll)
%SIMPLEX_KALMAN Perform a Simplex Kalman Filter Measurement Fusion Step
%  l measurement vector
%  H design matrix
%  x a priori state vector
%  Qxx a priori covariance matrix of state vector x
%  Qll covariance matrix of measurement vector l
% Return values:
%  Qxx a posteriori covariance matrix of state vector
%  dx a posteriori state vector
u = size(H,2); % unknowns in state vector
ltotal = [x;l];
Atotal = [eye(u);H];
Qtotal = blkdiag(Qxx, Qll);
[ldash, Adash, Lw] = homogenize(ltotal, Qtotal, Atotal);
[dx, res, info] = cl1norm(Adash, ldash); % Simplex Algorithm
if (info(1) ~= 0)
    error('simplex_kalman adjustment failed');
end
vdash = Adash*dx - ldash;
w = abs(vdash);
winv = w.^(-1);
numInf = 1e16; % Epsilon
winv(winv > numInf) = numInf;
W = diag(winv);
W2 = diag(winv.^(0.5));
Astar = W2*Adash;
Qllstar = W;
[Qstar, Rstar] = qr(Astar, 0);
RinvStar = Rstar\eye(u);
QxxL1 = RinvStar*Qstar'*Qllstar*Qstar*RinvStar';
Qxx = QxxL1;

end

function [ldash, Adash, Lw] = homogenize(l, Qll, A)
%HOMOGENIZE Compute a modified measurement vector l and a modified
% design matrix A, so that the covariance matrix Qll is not needed.
%
% For a least-squares problem:
% l = A*x
```

```
%
% ldash = Lw*l;
% Adash = Lw*A;
%
% with Qll as the covariance of l
% Qll = L*L' (Cholesky decomposition)
%
% P = inv(Qll)
%
% inv(Qll) = (L*L')^(-1) (i.e. inv(L*L'))
% inv(Qll) = (L')^(-1)*L^(-1)
% inv(Qll) = (L^(-1))'*L^(-1) = P
if nargin ~= 3
    error('Usage: [ldash, Adash, Lw] = homogenize(l, Qll, A).');
end

% cholesky decomposition:
L = chol(Qll, 'lower'); % L*L' = Qll

% mldivide checks for triangular matrices
n = size(Qll,1);
Lw = L\eye(n); % Lw = inv(L);

ldash = Lw*l;
Adash = Lw*A;

end
```

B Quellcode Simplex Linear Programming Solver

Der folgende Quellcode wurde im Rahmen dieser Arbeit für die Simplex $\mathscr{L}_1$ Lösung erarbeitet und verwendet. Der Algorithmus basiert auf der Veröffentlichung von Barrodale und Roberts (1980). Der Code wurde von FORTRAN in C++ übersetzt und so umgesetzt, dass er entweder als MATLAB MEX Modul nahtlos und sehr einfach in einer MATLAB Umgebung genutzt werden kann oder als *.cpp* Modul in einer C++ Anwendung. Dabei wird die EIGEN Mathebibliothek in der Version 3 genutzt (Guennebaud und Jacob, 2018).

```
/**
 * @file cl1norm.cpp
 *
 * SIMPLEX LINEAR PROGRAMMING SOLVER
 * =================================
 *
 * MEX Arguments: A, B, C, D, E, F
 *  A*X = B
 *  C*X = D
 *  E*X <= F
 *
 * Return value:
 *  X
 *  RESIDUALS (optional)
 *  INFO (optional)
 *
 * @brief
 * This function uses a modification of the simplex
 * method of linear programming to calculate an L1 solution
 * to a k-by-n system of linear equations
 *            A*X=B
 * subject to l linear equality constraints
 *            C*X=D
 * and m linear inequality constraints
 *            E*X <= F
 *
 * C/C++ code by Jan Zwiener (2015-2018).
 *
 * INFO vector:
 * INFO(1) = return code:
 *                      0- optimal solution found,
 *                      1- no feasible solution to the
 *                         constraints,
 *                      2- calculations terminated
```

```
 *                            prematurely due to rounding errors,
 *                        3- maximum number of iterations reached.
 *
 * INFO(2) = minimum sum of absolute values of the residuals.
 * INFO(3) = number of simplex iterations.
 *
 * Based on:
 *     I. Barrodale and F. D. K. Roberts. 1980.
 *     Algorithm 552:
 *     Solution of the Constrained I1 Linear Approximation Problem [F4].
 *     ACM Trans. Math. Softw. 6, 2 (June 1980), 231-235.
 *
 *     FORTRAN code:
 *     552.f -- translated by f2c (version 20100827).
 *
 * Compile in MATLAB as .mex file with:
 *      mex -O cl1norm.cpp
 * @{ */

/*****************************************************************************
 * INCLUDE FILES
 *****************************************************************************/

#ifdef MATLAB_MEX_FILE
  #include "mex.h"
#else
  #include <stdio.h> /* printf */
  #include <stdbool.h> /* bool type */
  #ifdef __cplusplus
    /* Eigen Math Lib 3.3 req. http://eigen.tuxfamily.org/index.php */
    #include <Eigen/Dense>
  #endif
#endif

#include <math.h> /* fabs() */
#include <stdlib.h> /* abs() */

/*****************************************************************************
 * DEFINES
 *****************************************************************************/

#define ENABLE_EIGEN_CPP_INTERFACE
#define DEFAULT_TOLERANCE (2e-11) /* change the MATLAB mexFunction() help text
                                     if you change this */

#define DEFAULT_TOLERANCE_FLOAT (1e-5)

/*****************************************************************************
 * TYPEDEFS
 *****************************************************************************/
```

```
/* some typedefs used in the cl1() function */
#ifdef MATLAB_MEX_FILE
typedef double real;
#else
using namespace Eigen;
typedef enum cl1_result_enum
{
    CL1_OPT_SOL_FOUND      = 0, /* optimal solution found */
    CL1_NO_SOL_FOUND       = 1, /* no feasible solution to the constraints, */
    CL1_ROUNDING_ERRORS    = 2, /* calculations terminated prematurely due to rounding
    errors */
    CL1_MAX_ITER           = 3, /* maximum number of iterations reached */
    CL1_MATRIX_DIM_INVALID = 4, /* maximum number of iterations reached */

    CL1_RESULT_MAX
} cl1_result_t;
#endif

/*****************************************************************************
 * LOCAL FUNCTION PROTOTYPES
 *****************************************************************************/

/*****************************************************************************
 * FUNCTION BODIES
 *****************************************************************************/

/* helper function to make sure the array access is not out of bound */
#ifdef NDEBUG
#define DBGCHECK(row, col, rows_total, cols_total, line) (0)
#else
static int DBGCHECK(int row, int col, int rows_total, int cols_total, int line)
{
    row--; col--;
    if (row >= rows_total || col >= cols_total || row < 0 || col < 0)
    {
        printf("Error in line %i: row %i col %i\n", line, row, col);
#ifdef MATLAB_MEX_FILE
        mexErrMsgIdAndTxt("JanZwiener:cl1norm:dimension",
                          "cl1norm internal error.");
#endif
    }
    return 0; /* always return 0 to be transparent to the MAT() macro */
}
#endif

/* helper macro for MATRIX access. This is using column-major access
 * for the MEX interface (MATLAB has Fortran roots).
 * The index is not zero-based! So it can be directly used in the
 * Fortran code.
```

```
 * Eigen is also using column-major access. But if you need row-major,
 * there is a macro for that too (see below).
 * DBGCHECK is here to make sure we don't mess something up.
 * DBGCHECK is only active if NDEBUG is not defined. */
#define MAT(M, row, column, rows_total, cols_total) \
    M[(((row)-1) + ((column)-1)*(rows_total)) + \
    DBGCHECK(row, column, rows_total, cols_total, __LINE__)] /* column-major */
/* row-major access: */
#if 0
#define MAT(M, row, column, rows_total, cols_total) \
    M[(((column)-1) + ((row)-1)*(cols_total)) + \
    DBGCHECK(row, column, rows_total, cols_total, __LINE__)]
#endif

/* THIS FUNCTION USES A MODIFICATION OF THE SIMPLEX
   METHOD OF LINEAR PROGRAMMING TO CALCULATE AN L1 SOLUTION
   TO A K BY N SYSTEM OF LINEAR EQUATIONS
                AX=B
   SUBJECT TO L LINEAR EQUALITY CONSTRAINTS
                CX=D
   AND M LINEAR INEQUALITY CONSTRAINTS
                EX.LE.F.
   DESCRIPTION OF PARAMETERS
   K       NUMBER OF ROWS OF THE MATRIX A (K.GE.1).
   L       NUMBER OF ROWS OF THE MATRIX C (L.GE.0).
   M       NUMBER OF ROWS OF THE MATRIX E (M.GE.0).
   N       NUMBER OF COLUMNS OF THE MATRICES A,C,E (N.GE.1).
   KLMD    SET TO AT LEAST K+L+M FOR ADJUSTABLE DIMENSIONS.
   KLM2D   SET TO AT LEAST K+L+M+2 FOR ADJUSTABLE DIMENSIONS.
   NKLMD   SET TO AT LEAST N+K+L+M FOR ADJUSTABLE DIMENSIONS.
   N2D     SET TO AT LEAST N+2 FOR ADJUSTABLE DIMENSIONS
   Q       TWO DIMENSIONAL REAL ARRAY WITH KLM2D ROWS AND
           AT LEAST N2D COLUMNS.
           ON ENTRY THE MATRICES A,C AND E, AND THE VECTORS
           B,D AND F MUST BE STORED IN THE FIRST K+L+M ROWS
           AND N+1 COLUMNS OF Q AS FOLLOWS
                A B
            Q = C D
                E F
           THESE VALUES ARE DESTROYED BY THE SUBROUTINE.
   KODE    A CODE USED ON ENTRY TO, AND EXIT
           FROM, THE SUBROUTINE.
           ON ENTRY, THIS SHOULD NORMALLY BE SET TO 0.
           HOWEVER, IF CERTAIN NONNEGATIVITY CONSTRAINTS
           ARE TO BE INCLUDED IMPLICITLY, RATHER THAN
           EXPLICITLY IN THE CONSTRAINTS EX.LE.F, THEN KODE
           SHOULD BE SET TO 1, AND THE NONNEGATIVITY
           CONSTRAINTS INCLUDED IN THE ARRAYS X AND
           RES (SEE BELOW).
           ON EXIT, KODE HAS ONE OF THE
```

```
          FOLLOWING VALUES
               0- OPTIMAL SOLUTION FOUND,
               1- NO FEASIBLE SOLUTION TO THE
                  CONSTRAINTS,
               2- CALCULATIONS TERMINATED
                  PREMATURELY DUE TO ROUNDING ERRORS,
               3- MAXIMUM NUMBER OF ITERATIONS REACHED.
   TOLER  A SMALL POSITIVE TOLERANCE. EMPIRICAL
          EVIDENCE SUGGESTS TOLER = 10**(-D*2/3),
          WHERE D REPRESENTS THE NUMBER OF DECIMAL
          DIGITS OF ACCURACY AVAILABLE. ESSENTIALLY,
          THE SUBROUTINE CANNOT DISTINGUISH BETWEEN ZERO
          AND ANY QUANTITY WHOSE MAGNITUDE DOES NOT EXCEED
          TOLER. IN PARTICULAR, IT WILL NOT PIVOT ON ANY
          NUMBER WHOSE MAGNITUDE DOES NOT EXCEED TOLER.
   ITER   ON ENTRY ITER MUST CONTAIN AN UPPER BOUND ON
          THE MAXIMUM NUMBER OF ITERATIONS ALLOWED.
          A SUGGESTED VALUE IS 10*(K+L+M). ON EXIT ITER
          GIVES THE NUMBER OF SIMPLEX ITERATIONS.
   X      ONE DIMENSIONAL REAL ARRAY OF SIZE AT LEAST N2D.
          ON EXIT THIS ARRAY CONTAINS A
          SOLUTION TO THE L1 PROBLEM. IF KODE=1
          ON ENTRY, THIS ARRAY IS ALSO USED TO INCLUDE
          SIMPLE NONNEGATIVITY CONSTRAINTS ON THE
          VARIABLES. THE VALUES -1, 0, OR 1
          FOR X(J) INDICATE THAT THE J-TH VARIABLE
          IS RESTRICTED TO BE .LE.0, UNRESTRICTED,
          OR .GE.0 RESPECTIVELY.
   RES    ONE DIMENSIONAL REAL ARRAY OF SIZE AT LEAST KLMD.
          ON EXIT THIS CONTAINS THE RESIDUALS B-AX
          IN THE FIRST K COMPONENTS, D-CX IN THE
          NEXT L COMPONENTS (THESE WILL BE =0),AND
          F-EX IN THE NEXT M COMPONENTS. IF KODE=1 ON
          ENTRY, THIS ARRAY IS ALSO USED TO INCLUDE SIMPLE
          NONNEGATIVITY CONSTRAINTS ON THE RESIDUALS
          B-AX. THE VALUES -1, 0, OR 1 FOR RES(I)
          INDICATE THAT THE I-TH RESIDUAL (1.LE.I.LE.K) IS
          RESTRICTED TO BE .LE.0, UNRESTRICTED, OR .GE.0
          RESPECTIVELY.
   ERROR  ON EXIT, THIS GIVES THE MINIMUM SUM OF
          ABSOLUTE VALUES OF THE RESIDUALS.
   CU     A TWO DIMENSIONAL REAL ARRAY WITH TWO ROWS AND
          AT LEAST NKLMD COLUMNS USED FOR WORKSPACE.
   IU     A TWO DIMENSIONAL INTEGER ARRAY WITH TWO ROWS AND
          AT LEAST NKLMD COLUMNS USED FOR WORKSPACE.
   S      INTEGER ARRAY OF SIZE AT LEAST KLMD, USED FOR
          WORKSPACE.
*/
#ifndef MATLAB_MEX_FILE
template <typename real>
```

```
#endif
static void cl1(const int *k, const int *l, const int *m,
                const int *n, const int *klmd, const int *klm2d,
                const int *nklmd, const int *n2d, real *q,
                int *kode, const real *toler, int *iter, real *x,
                real *res, real *error, real *cu, int *iu, int *s)
{
    int i, j;
    real z__;
    int n1, n2, ia, ii, kk, in, nk, js;
    real sn, zu, zv;
    int nk1, klm, jmn, nkl, jpn;
    real cuv;
    double sum;
    int klm1, klm2, nkl1, iimn, nklm;
    real xmin, xmax;
    int iout, iineg, maxit;
    real pivot;
    int iphase, kforce;
    real tpivot;
    const int qrows = *klm2d;
    const int qcols = *n2d;

    /* INITIALIZATION. */
    maxit = *iter;
    n1 = *n + 1;
    n2 = *n + 2;
    nk = *n + *k;
    nk1 = nk + 1;
    nkl = nk + *l;
    nkl1 = nkl + 1;
    klm = *k + *l + *m;
    klm1 = klm + 1;
    klm2 = klm + 2;
    nklm = *n + klm;
    kforce = 1;
    *iter = 0;
    js = 1;
    ia = 0;
    /* SET UP LABELS IN Q. */
    for (j = 1; j <= *n; ++j) {
        MAT(q, klm2, j, qrows, qcols) = (real)j;
    }
    for (i = 1; i <= klm; ++i) {
        MAT(q, i, n2, qrows, qcols) = (real)(*n + i);
        if (MAT(q, i, n1, qrows, qcols) >= 0.0) {
            goto L30;
        }
        for (j = 1; j <= n2; ++j) {
            MAT(q, i, j, qrows, qcols) = -MAT(q, i, j, qrows, qcols);
```

```
        }
    L30:
        ;
    }
    /* SET UP PHASE 1 COSTS. */
    iphase = 2;
    for (j = 1; j <= nklm; ++j) {
        MAT(cu, 1, j, 2, *nklmd) = (real)0.0;
        MAT(cu, 2, j, 2, *nklmd) = (real)0.0;
        MAT(iu, 1, j, 2, *nklmd) = 0;
        MAT(iu, 2, j, 2, *nklmd) = 0;
    }
    if (*l == 0) {
        goto L60;
    }
    for (j = nk1; j <= nkl; ++j) {
        MAT(cu, 1, j, 2, *nklmd) = (real)1.0;
        MAT(cu, 2, j, 2, *nklmd) = (real)1.0;
        MAT(iu, 1, j, 2, *nklmd) = 1;
        MAT(iu, 2, j, 2, *nklmd) = 1;
    }
    iphase = 1;
L60:
    if (*m == 0) {
        goto L80;
    }
    for (j = nkl1; j <= nklm; ++j) {
        MAT(cu, 2, j, 2, *nklmd) = (real)1.0;
        MAT(iu, 2, j, 2, *nklmd) = 1;
        jmn = j - *n;
        if (MAT(q, jmn, n2, qrows, qcols) < 0) {
            iphase = 1;
        }
        /* L70: */
    }
L80:
    if (*kode == 0) {
        goto L150;
    }
    for (j = 1; j <= *n; ++j) {
        if (x[j-1] < 0.0) {
            goto L90;
        }
        else if (x[j-1] == 0) {
            goto L110;
        }
        else {
            goto L100;
        }
    L90:
```

```
        MAT(cu, 1, j, 2, *nklmd) = (real)1.0;
        MAT(iu, 1, j, 2, *nklmd) = 1;
        goto L110;
    L100:
        MAT(cu, 2, j, 2, *nklmd) = (real)1.0;
        MAT(iu, 2, j, 2, *nklmd) = 1;
    L110:
        ;
    }
    for (j = 1; j <= *k; ++j) {
        jpn = j + *n;
        if (res[j-1] < 0) {
            goto L120;
        }
        else if (res[j-1] == 0) {
            goto L140;
        }
        else {
            goto L130;
        }
    L120:
        MAT(cu, 1, jpn, 2, *nklmd) = (real)1.0;
        MAT(iu, 1, jpn, 2, *nklmd) = 1;
        if (MAT(q, j, n2, qrows, qcols) > 0) {
            iphase = 1;
        }
        goto L140;
    L130:
        MAT(cu, 2, jpn, 2, *nklmd) = (real)1.0;
        MAT(iu, 2, jpn, 2, *nklmd) = 1;
        if (MAT(q, j, n2, qrows, qcols) < 0) {
            iphase = 1;
        }
    L140:
        ;
    }
L150:
    if (iphase == 2) {
        goto L500;
    }
    /* COMPUTE THE MARGINAL COSTS. */
L160:
    for (j = js; j <= n1; ++j) {
        sum = 0;
        for (i = 1; i <= klm; ++i) {
            ii = (int)MAT(q, i, n2, qrows, qcols);
            if (ii < 0) {
                goto L170;
            }
            z__ = MAT(cu, 1, ii, 2, *nklmd);
```

```
            goto L180;
        L170:
            iineg = -ii;
            z__ = MAT(cu, 2, iineg, 2, *nklmd);
        L180:
            sum += (double)MAT(q, i, j, qrows, qcols) * (double)z__;
        }
        MAT(q, klm1, j, qrows, qcols) = (real)sum;
    }
    for (j = js; j <= *n; ++j) {
        ii = (int)MAT(q, klm2, j, qrows, qcols);
        if (ii < 0) {
            goto L210;
        }
        z__ = MAT(cu, 1, ii, 2, *nklmd);
        goto L220;
    L210:
        iineg = -ii;
        z__ = MAT(cu, 2, iineg, 2, *nklmd);
    L220:
        MAT(q, klm1, j, qrows, qcols) -= z__;
    }
    /* DETERMINE THE VECTOR TO ENTER THE BASIS. */
L240:
    xmax = 0.0;
    if (js > *n) {
        goto L490;
    }
    for (j = js; j <= *n; ++j) {
        zu = MAT(q, klm1, j, qrows, qcols);
        ii = (int)MAT(q, klm2, j, qrows, qcols);
        if (ii > 0) {
            goto L250;
        }
        ii = -ii;
        zv = zu;
        zu = -zu - MAT(cu, 1, ii, 2, *nklmd) - MAT(cu, 2, ii, 2, *nklmd);
        goto L260;
    L250:
        zv = -zu - MAT(cu, 1, ii, 2, *nklmd) - MAT(cu, 2, ii, 2, *nklmd);
    L260:
        if (kforce == 1 && ii > *n) {
            goto L280;
        }
        if (MAT(iu, 1, ii, 2, *nklmd) == 1) {
            goto L270;
        }
        if (zu <= xmax) {
            goto L270;
        }
```

```
        xmax = zu;
        in = j;
    L270:
        if (MAT(iu, 2, ii, 2, *nklmd) == 1) {
            goto L280;
        }
        if (zv <= xmax) {
            goto L280;
        }
        xmax = zv;
        in = j;
    L280:
        ;
    }
    if (xmax <= *toler) {
        goto L490;
    }
    if (MAT(q, klm1, in, qrows, qcols) == xmax) {
        goto L300;
    }
    for (i = 1; i <= klm2; ++i) {
        MAT(q, i, in, qrows, qcols) *= (real)(-1.0);
    }
    MAT(q, klm1, in, qrows, qcols) = xmax;
    /* DETERMINE THE VECTOR TO LEAVE THE BASIS. */
L300:
    if (iphase == 1 || ia == 0) {
        goto L330;
    }
    xmax = 0.0;
    for (i = 1; i <= ia; ++i) {
        z__ = fabs(MAT(q, i, in, qrows, qcols));
        if (z__ <= xmax) {
            goto L310;
        }
        xmax = z__;
        iout = i;
    L310:
        ;
    }
    if (xmax <= *toler) {
        goto L330;
    }
    for (j = 1; j <= n2; ++j) {
        z__ = MAT(q, ia, j, qrows, qcols);
        MAT(q, ia, j, qrows, qcols) = MAT(q, iout, j, qrows, qcols);
        MAT(q, iout, j, qrows, qcols) = z__;
    }
    iout = ia;
    --ia;
```

```
    pivot = MAT(q, iout, in, qrows, qcols);
    goto L420;
L330:
    kk = 0;
    for (i = 1; i <= klm; ++i) {
        z__ = MAT(q, i, in, qrows, qcols);
        if (z__ <= *toler) {
            goto L340;
        }
        ++kk;
        res[kk-1] = MAT(q, i, n1, qrows, qcols) / z__;
        s[kk-1] = i;
    L340:
        ;
    }
L350:
    if (kk > 0) {
        goto L360;
    }
    *kode = 2;
    goto L590;
L360:
    xmin = res[0];
    iout = s[0];
    j = 1;
    if (kk == 1) {
        goto L380;
    }
    for (i = 2; i <= kk; ++i) {
        if (res[i-1] >= xmin) {
            goto L370;
        }
        j = i;
        xmin = res[i-1];
        iout = s[i-1];
    L370:
        ;
    }
    res[j-1] = res[kk-1];
    s[j-1] = s[kk-1];
L380:
    --kk;
    pivot = MAT(q, iout, in, qrows, qcols);
    ii = (int)MAT(q, iout, n2, qrows, qcols);
    if (iphase == 1) {
        goto L400;
    }
    if (ii < 0) {
        goto L390;
    }
```

```
    if (MAT(iu, 2, ii, 2, *nklmd) == 1) {
        goto L420;
    }
    goto L400;
L390:
    iineg = -ii;
    if (MAT(iu, 1, iineg, 2, *nklmd) == 1) {
        goto L420;
    }
L400:
    ii = abs(ii);
    cuv = MAT(cu, 1, ii, 2, *nklmd) + MAT(cu, 2, ii, 2, *nklmd);
    if (MAT(q, klm1, in, qrows, qcols) - pivot * cuv <= *toler) {
        goto L420;
    }
    /* BYPASS INTERMEDIATE VERTICES. */
    for (j = js; j <= n1; ++j) {
        z__ = MAT(q, iout, j, qrows, qcols);
        MAT(q, klm1, j, qrows, qcols) -= z__ * cuv;
        MAT(q, iout, j, qrows, qcols) = -z__;
        /* L410: */
    }
    MAT(q, iout, n2, qrows, qcols) *= (real)(-1.0);
    goto L350;
    /* GAUSS-JORDAN ELIMINATION. */
L420:
    if (*iter < maxit) {
        goto L430;
    }
    *kode = 3;
    goto L590;
L430:
    ++(*iter);
    for (j = js; j <= n1; ++j) {
        if (j != in) {
            MAT(q, iout, j, qrows, qcols) /= pivot;
        }
    }
    for (j = js; j <= n1; ++j) {
        if (j == in) {
            goto L460;
        }
        z__ = -MAT(q, iout, j, qrows, qcols);
        for (i = 1; i <= klm1; ++i) {
            if (i != iout) {
                MAT(q, i, j, qrows, qcols) += z__ * MAT(q, i, in, qrows, qcols);
            }
        }
    L460:
        ;
```

```
    }
    tpivot = -pivot;
    for (i = 1; i <= klm1; ++i) {
        if (i != iout) {
            MAT(q, i, in, qrows, qcols) /= tpivot;
        }
    }
    MAT(q, iout, in, qrows, qcols) = (real)(1.0) / pivot;
    z__ = MAT(q, iout, n2, qrows, qcols);
    MAT(q, iout, n2, qrows, qcols) = MAT(q, klm2, in, qrows, qcols);
    MAT(q, klm2, in, qrows, qcols) = z__;
    ii = (int)fabs(z__);
    if (MAT(iu, 1, ii, 2, *nklmd) == 0 || MAT(iu, 2, ii, 2, *nklmd) == 0) {
        goto L240;
    }
    for (i = 1; i <= klm2; ++i) {
        z__ = MAT(q, i, in, qrows, qcols);
        MAT(q, i, in, qrows, qcols) = MAT(q, i, js, qrows, qcols);
        MAT(q, i, js, qrows, qcols) = z__;
    }
    ++js;
    goto L240;
    /* TEST FOR OPTIMALITY. */
L490:
    if (kforce == 0) {
        goto L580;
    }
    if (iphase == 1 && MAT(q, klm1, n1, qrows, qcols) <= *toler) {
        goto L500;
    }
    kforce = 0;
    goto L240;
    /* SET UP PHASE 2 COSTS. */
L500:
    iphase = 2;
    for (j = 1; j <= nklm; ++j) {
        MAT(cu, 1, j, 2, *nklmd) = 0;
        MAT(cu, 2, j, 2, *nklmd) = 0;
    }
    for (j = n1; j <= nk; ++j) {
        MAT(cu, 1, j, 2, *nklmd) = (real)1.0;
        MAT(cu, 2, j, 2, *nklmd) = (real)1.0;
    }
    for (i = 1; i <= klm; ++i) {
        ii = (int)MAT(q, i, n2, qrows, *nklmd);
        if (ii > 0) {
            goto L530;
        }
        ii = -ii;
        if (MAT(iu, 2, ii, 2, *nklmd) == 0) {
```

```
            goto L560;
        }
        MAT(cu, 2, ii, 2, *nklmd) = 0;
        goto L540;
    L530:
        if (MAT(iu, 1, ii, 2, *nklmd) == 0) {
            goto L560;
        }
        MAT(cu, 1, ii, 2, *nklmd) = 0;
    L540:
        ++ia;
        for (j = 1; j <= n2; ++j) {
            z__ = MAT(q, ia, j, qrows, qcols);
            MAT(q, ia, j, qrows, qcols) = MAT(q, i, j, qrows, qcols);
            MAT(q, i, j, qrows, qcols) = z__;
        }
    L560:
        ;
    }
    goto L160;
L570:
    if (MAT(q, klm1, n1, qrows, qcols) <= *toler) {
        goto L500;
    }
    *kode = 1;
    goto L590;
L580:
    if (iphase == 1) {
        goto L570;
    }
    /* PREPARE OUTPUT. */
    *kode = 0;
L590:
    sum = 0.;
    for (j = 1; j <= *n; ++j) {
        x[j-1] = 0;
    }
    for (i = 1; i <= klm; ++i) {
        res[i-1] = 0;
    }
    for (i = 1; i <= klm; ++i) {
        ii = (int)MAT(q, i, n2, qrows, qcols);
        sn = 1.0;
        if (ii > 0) {
            goto L620;
        }
        ii = -ii;
        sn = (real)(-1.0);
    L620:
        if (ii > *n) {
```

```
            goto L630;
        }
        x[ii-1] = sn * MAT(q, i, n1, qrows, qcols);
        goto L640;
    L630:
        iimn = ii - *n;
        res[iimn-1] = sn * MAT(q, i, n1, qrows, qcols);
        if (ii >= n1 && ii <= nk) {
            sum += (double)MAT(q, i, n1, qrows, qcols);
        }
    L640:
        ;
    }
    *error = (real)sum;
}

#ifndef MATLAB_MEX_FILE
void cl1_c_float(const int *k, const int *l, const int *m,
                 const int *n, const int *klmd, const int *klm2d,
                 const int *nklmd, const int *n2d, float *q,
                 int *kode, const float *toler, int *iter, float *x,
                 float *res, float *error, float *cu, int *iu, int *s)
{
    cl1<float>(k, l, m, n, klmd, klm2d, nklmd, n2d,
               q, kode, toler, iter, x, res, error, cu, iu, s);
}

void cl1_c_double(const int *k, const int *l, const int *m,
                  const int *n, const int *klmd, const int *klm2d,
                  const int *nklmd, const int *n2d, double *q,
                  int *kode, const double *toler, int *iter, double *x,
                  double *res, double *error, double *cu, int *iu, int *s)
{
    cl1<double>(k, l, m, n, klmd, klm2d, nklmd, n2d,
                q, kode, toler, iter, x, res, error, cu, iu, s);
}
#endif

/*****************************************************************************
 * EIGEN MATH LIBRARY INTERFACE (optional)
 *****************************************************************************/

#ifndef MATLAB_MEX_FILE

#ifdef ENABLE_EIGEN_CPP_INTERFACE
template <typename T>
static cl1_result_t cl1_eigen(
    const Matrix<T, Dynamic, Dynamic>& A,
    const Matrix<T, Dynamic, 1>& B,
    Matrix<T, Dynamic, 1>& X,
```

```
    const Matrix<T, Dynamic, Dynamic>* C,
    const Matrix<T, Dynamic, 1>* D,
    const Matrix<T, Dynamic, Dynamic>* E,
    const Matrix<T, Dynamic, 1>* F,
    T tolerance,
    int maxiter,
    Matrix<T, Dynamic, 1>* residuals)
{
    if (tolerance <= 0)
    {
        if (sizeof(T) == sizeof(float))
        {
            tolerance = (T)DEFAULT_TOLERANCE_FLOAT;
        }
        else
        {
            tolerance = (T)DEFAULT_TOLERANCE;
        }
    }

    const int k = (int)A.rows();
    const int n = (int)A.cols();
    const int l = (C && D) ? (int)C->rows() : 0;
    const int m = (E && F) ? (int)E->rows() : 0;
    const int klmd = k + l + m;
    const int klm2d = k + l + m + 2;
    const int nklmd = n + k + l + m + 2;
    const int n2d = n + 2;

    if ((k < 1) || (n < 1) || (B.rows() != k))
    {
        return CL1_MATRIX_DIM_INVALID;
    }
    if (C && D)
    {
        if (C->rows() != D->rows() || C->rows() < 1 ||
            C->cols() != n || D->cols() != 1)
        {
            return CL1_MATRIX_DIM_INVALID;
        }
    }
    if (E && F)
    {
        if (E->rows() != F->rows() || E->rows() < 1 ||
            E->cols() != n || F->cols() != 1)
        {
            return CL1_MATRIX_DIM_INVALID;
        }
    }
```

```
if (maxiter <= 0)
{
    maxiter = 10*(k+l+m); /* default max. iterations */
}

Matrix<T, Dynamic, Dynamic> Q(klm2d, n2d);
Matrix<T, Dynamic, Dynamic> X2d(n2d, 1); /* unknown vector for cl1() */

Q.setZero();
Q.block(0,   0, k, n) = A;
Q.block(0,   n, k, 1) = B;
if (C && D)
{
    Q.block(k, 0, l, n) = *C;
    Q.block(k, n, l, 1) = *D;
}
if (E && F)
{
    Q.block(k + l, 0, m, n) = *E;
    Q.block(k + l, n, m, 1) = *F;
}

Matrix<T, Dynamic, Dynamic> R(klmd, 1); /* residuals */
Matrix<T, 2, Dynamic> CU(2, nklmd);
Matrix<int, 2, Dynamic> IU(2, nklmd);
Matrix<int, Dynamic, 1> S(klmd, 1);

X2d.setZero();
R.setZero();
CU.setZero();
IU.setZero();
S.setZero();

int kode = 0;
T error;

cl1<T>(&k,
       &l,
       &m,
       &n,
       &klmd,
       &klm2d,
       &nklmd,
       &n2d,
       Q.data(),
       &kode,
       &tolerance,
       &maxiter,
       X2d.data(),
       R.data(),
```

```
            &error,
            CU.data(),
            IU.data(),
            S.data());

    if (residuals)
    {
        *residuals = R;
    }
    X = X2d.block(0, 0, n, 1);

    return (cl1_result_t)kode;
}

cl1_result_t cl1_float(
    const Eigen::Matrix<float, Eigen::Dynamic, Eigen::Dynamic>& A,
    const Eigen::Matrix<float, Eigen::Dynamic, 1>& B,
    Eigen::Matrix<float, Eigen::Dynamic, 1>& X,
    const Eigen::Matrix<float, Eigen::Dynamic, Eigen::Dynamic>* C,
    const Eigen::Matrix<float, Eigen::Dynamic, 1>* D,
    const Eigen::Matrix<float, Eigen::Dynamic, Eigen::Dynamic>* E,
    const Eigen::Matrix<float, Eigen::Dynamic, 1>* F,
    float tolerance,
    int maxiter,
    Eigen::Matrix<float, Eigen::Dynamic, 1>* residuals)
{
    return cl1_eigen<float>(A, B, X, C, D, E, F, tolerance, maxiter, residuals);
}

cl1_result_t cl1_double(
    const Matrix<double, Dynamic, Dynamic>& A,
    const Matrix<double, Dynamic, 1>& B,
    Matrix<double, Dynamic, 1>& X,
    const Matrix<double, Dynamic, Dynamic>* C,
    const Matrix<double, Dynamic, 1>* D,
    const Matrix<double, Dynamic, Dynamic>* E,
    const Matrix<double, Dynamic, 1>* F,
    double tolerance,
    int maxiter,
    Matrix<double, Dynamic, 1>* residuals)
{
    return cl1_eigen<double>(A, B, X, C, D, E, F, tolerance, maxiter, residuals);
}
#endif /* ENABLE_EIGEN_CPP_INTERFACE */

#endif

/*****************************************************************************
 * MATLAB MEX INTERFACE (optional)
 *****************************************************************************/
```

```
#ifdef MATLAB_MEX_FILE

/* Helper function to copy matrix "in" to matrix "out"
 * at position out_pos_m,n.
 *
 * MATLAB syntax would be:
 * out(out_pos_m:out_pos_m+in_m, out_pos_n:out_pos_n+in_n) = in;
 *
 * in: input matrix
 * in_m: number of rows in "in"
 * in_n: number of columns in "in"
 * out: pointer to output matrix
 * out_m: number of rows in "out"
 * out_n: number of columns in "out"
 * out_pos_m: target location in out
 * out_pos_n: target location in out
 */
static void copymatrix(const double* in, int in_m, int in_n,
                       double* out, int out_m, int out_n,
                       int out_pos_m, int out_pos_n)
{
    if (in_m == 0 || in_n == 0 || in == NULL)
    {
        return; /* ignore empty input matrices */
    }

    /* don't ignore broken input */
    if (out_pos_m + in_m > out_m || out_pos_n + in_n > out_n ||
        in_m > out_m || in_n > out_n || in_m < 0 || in_n < 0)
    {
        mexErrMsgIdAndTxt("JanZwiener:cl1norm:dimension",
                          "Matrix size mismatch");
        return;
    }

    int i, j;
    for (j=0;j<in_n;j++) /* for each column */
    {
        for (i=0;i<in_m;i++) /* for each row */
        {
            MAT(out, i+out_pos_m+1, j+out_pos_n+1, out_m, out_n) =
                MAT(in, i+1, j+1, in_m, in_n);
        }
    }
}

/* The gateway function */
void mexFunction(int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[])
{
```

```
const int min_input = 2;
real toler = (real)DEFAULT_TOLERANCE;
int kode = 0; /* return value from cl1() function */
int iter; /* max iterations for cl1.  */
real err; /* minimum sum of residuals */
int i;

// 3 arguments are not valid (if C is given, D is also required).
// 5 arguments are not valid (if E is given, F is also required).
if(nrhs < min_input || nrhs == 3 || nrhs == 5) {
    mexErrMsgIdAndTxt(
    "JanZwiener:cl1norm:nrhs",
    "Linear Programming Simplex Solver.\n"
    "Usage: [x, res, info] = cl1norm(A, B, C, D, E, F, tol, maxiter);\n"
    "\n"
    "Input: A, B, C (optional), D (optional),"
    " E (optional), F (optional),\n"
    "tolerance (optional), maxiter (optional).\n"
    "A*x = B, C*x = D, E*x <= F\n"
    "C and D can be empty matrices [] and/or E and F"
    " can be empty matrices [].\n"
    "tolerance: A small positive tolerance. Default: 1e-9.\n"
    "maxiter: Maximum number of iterations for the algorithm.\n"
    "Output: x, residuals (optional), simplexinfo (optional)\n"
    "simplexinfo(1): 0 - optimal solution found. >=1 no solution found.\n"
    "simplexinfo(2): Minimum sum of absolute values of the residuals.\n"
    "simplexinfo(3): Number of simplex iterations.\n"
    "");
}

if(nlhs < 1) {
    mexErrMsgIdAndTxt("JanZwiener:cl1norm:nlhs",
                      "One output required.");
}

/* make sure the input arguments are real */
for (i=0;i<nrhs;i++)
{
    if( !mxIsDouble(prhs[i]) || mxIsComplex(prhs[i]) )
    {
        mexErrMsgIdAndTxt("JanZwiener:cl1norm:notReal",
                          "Input must be real.");
    }
}

const mxArray* mA = prhs[0];
const mxArray* mB = prhs[1];
const mxArray* mC = (nrhs >= 4) ? prhs[2] : NULL;
const mxArray* mD = (nrhs >= 4) ? prhs[3] : NULL;
const mxArray* mE = (nrhs >= 6) ? prhs[4] : NULL;
```

```
    const mxArray* mF = (nrhs >= 6) ? prhs[5] : NULL;

    int k = (int)mxGetM(mA);
    int n = (int)mxGetN(mA);
    int l = 0;
    int m = 0;

    if (nrhs >= 4)
    {
        l = (int)mxGetM(mC);
    }
    if (nrhs >= 6)
    {
        m = (int)mxGetM(mE);
    }
    if (nrhs >= 7)
    {
        toler = mxGetPr(prhs[6])[0];
    }
    if (nrhs >= 8)
    {
        iter = (int)mxGetPr(prhs[7])[0];
    }
    else
    {
        iter = 10*(k+l+m); /* default max. iterations */
    }

    const int klmd = k + l + m;
    const int klm2d = k + l + m + 2;
    const int nklmd = n + k + l + m + 2;
    const int n2d = n + 2;

    if (k < 1 || n < 1)
    {
        mexErrMsgIdAndTxt("JanZwiener:cl1norm:dimension",
                          "Matrix A empty.");
    }

    if ((int)mxGetM(mB) != k ||
        (int)mxGetN(mB) != 1)
    {
        mexErrMsgIdAndTxt("JanZwiener:cl1norm:dimension",
                          "B matrix dimension does not match A matrix.");
    }

    if (l > 0)
    {
        if ((int)mxGetN(mC) != n ||
            (int)mxGetM(mD) != l ||
```

```
        (int)mxGetN(mD) != 1)
    {
        mexErrMsgIdAndTxt("JanZwiener:cl1norm:dimension",
                          "Input dimension mismatch.");
    }
}

if (m > 0)
{
    if ((int)mxGetN(mE) != n ||
        (int)mxGetM(mF) != m)
    {
        mexErrMsgIdAndTxt("JanZwiener:cl1norm:dimension",
                          "Input dimension mismatch.");
    }
}

// prepare input data
mxArray* mQ      = mxCreateDoubleMatrix(klm2d, n2d,   mxREAL);
mxArray* mX      = mxCreateDoubleMatrix(n2d,   1,     mxREAL);
mxArray* mXsmall = mxCreateDoubleMatrix(n,     1,     mxREAL);
mxArray* mRES    = mxCreateDoubleMatrix(klmd,  1,     mxREAL);
mxArray* mCU     = mxCreateDoubleMatrix(2,     nklmd, mxREAL);
int* AUI         = (int*)mxMalloc(2*nklmd*sizeof(int));
int* outS        = (int*)mxMalloc(klmd*sizeof(int));
double* AQ       = mxGetPr(mQ);
double* AX       = mxGetPr(mX);
double* AXsmall  = mxGetPr(mXsmall);
double* ARES     = mxGetPr(mRES);
double* ACU      = mxGetPr(mCU);
const double* AA = mxGetPr(mA);
const double* AB = mxGetPr(mB);
const double* AC = mC ? mxGetPr(mC) : NULL;
const double* AD = mD ? mxGetPr(mD) : NULL;
const double* AE = mE ? mxGetPr(mE) : NULL;
const double* AF = mF ? mxGetPr(mF) : NULL;

// copy all the input matrices in the big Q matrix.
//     +-----+
// Q = | A B |
//     | C D |
//     | E F |
//     |     |
//     +-----+
copymatrix(AA, k, n, AQ, klm2d, n2d, 0,   0);
copymatrix(AB, k, 1, AQ, klm2d, n2d, 0,   n);
copymatrix(AC, l, n, AQ, klm2d, n2d, k,   0);
copymatrix(AD, l, 1, AQ, klm2d, n2d, k,   n);
copymatrix(AE, m, n, AQ, klm2d, n2d, k+l, 0);
copymatrix(AF, m, 1, AQ, klm2d, n2d, k+l, n);
```

```
    // actual calculation
    cl1(&k, &l, &m, &n, &klmd, &klm2d, &nklmd, &n2d,
        AQ, &kode, &toler, &iter, AX, ARES, &err, ACU, AUI, outS);

    /* X has additional fields for the cl1 function. we only need
     * the first n-values */
    copymatrix(AX, n, 1, AXsmall, n, 1, 0, 0);
    plhs[0] = mXsmall;

    /* return residuals? */
    if (nlhs > 1)
    {
        plhs[1] = mRES;
    }
    else
    {
        mxDestroyArray(mRES);
    }

    /* return dbg info? */
    if (nlhs > 2)
    {
        mxArray* mSIMPLEXINFO = mxCreateDoubleMatrix(3, 1, mxREAL);
        double* ASIMPLEXINFO  = mxGetPr(mSIMPLEXINFO);
        ASIMPLEXINFO[0] = (double)kode;
        ASIMPLEXINFO[1] = (double)err;
        ASIMPLEXINFO[2] = (double)iter;

        plhs[2] = mSIMPLEXINFO;
    }

    // cleanup
    mxDestroyArray(mX);
    mxDestroyArray(mQ);
    mxDestroyArray(mCU);
    mxFree(AUI);
    mxFree(outS);
}

#endif /* MATLAB_MEX_FILE */

/* @} */
```

C Quellcode Savitzky-Golay Koeffizienten

```
function [c] = savgol(m, nl, nr, ld)
% SAVGOL  Calculate Savitzky-Golay filter coefficients (Jan Zwiener).
%   c = savgol(m,nl,nr,ld)
%   m is the order of the smoothing polynomial (positive integer)
%   nl number of left (past) data points (positive integer)
%   nr number of right (future) data points (positive integer)
%   ld is the order of the derivative desired. ld=0 for smoothing,
%     ld=1 for the filtered first derivate (needs to be divided by the
%     step size).
%
% Example cubic smoothing:
%   f = [1 1.4 1.5 1.4 1.3]'; % example data
%   c = savgol(3, 2, 2, 0);   % smooth in the middle with cubic polynomial
%   f_smooth = c'*f           % smoothed function value at index = 3
%
% Example calculate first derivative with 2nd order polynomial:
%   f  = [1 1.4 1.5 1.4 1]'; % example data
%   dt = 1.0;                % example data step size
%   c1 = savgol(2, 4, 0, 1); % 1st derivate at rightmost position
%   df_dt = (c1'*f)/dt;      % divide by step size

assert(nl >= 0 && nr >= 0 && ld <= m && nl+nr >= m && ld >= 0);
assert(mod(nl,1)==0 && mod(nr,1)==0 && mod(ld,1)==0 && mod(m,1)==0);

i = (-nl:nr)';   % rows of Vandermonde matrix
j = 0:m;         % columns of Vandermonde matrix
A = i .^ j;      % set up Vandermonde design matrix: A(line, col) = i^j;
K = (A'*A)\A';   % least squares: (A'*A)^(-1)*A'
c = K(ld+1, :)'; % line 1 = smoothing, line 2 = 1st derivative ...

end
```

Literaturverzeichnis

Abdelmalek, N. (1980). L_1 solution of overdetermined systems of linear equations, *ACM Transactions on Mathematical Software (TOMS)* **6**(2): 220–227. ACM.

Abdelmalek, N. und Malek, W. A. (2008). *Numerical Linear Approximation in C*, CRC Press.

Aggarwal, P., Syed, Z. und El-Sheimy, N. (2008). Thermal calibration of low cost MEMS sensors for land vehicle navigation system, *Vehicular Technology Conference, 2008. VTC Spring 2008.*, IEEE, pp. 2859–2863.

Ahlstrom, K., Torin, J., Fersán, K. und Nobrant, P. (2002). Redundancy management in distributed flight control systems: experience & simulations, *Digital Avionics Systems Conference, 2002. Proceedings. The 21st*, Vol. 2, IEEE, pp. 13C3–13C3.

Allan, D. W. (1966). Statistics of atomic frequency standards, *Proceedings of the IEEE* **54**(2): 221–230.

Allan, D. W., Ashby, N. und Hodge, C. C. (1997). *The science of timekeeping*, Hewlett-Packard.

Anderson, E., Bai, Z., Bischof, C., Blackford, S., Demmel, J., Dongarra, J., Du Croz, J., Greenbaum, A., Hammarling, S., McKenney, A. und Sorensen, D. (1999). *LAPACK Users' Guide*, third edn, Society for Industrial and Applied Mathematics, Philadelphia, PA. ISBN: 0-89871-447-8.

Arasaratnam, I. und Haykin, S. (2009). Cubature kalman filters, *IEEE Transactions on automatic control* **54**(6): 1254–1269.

Argentim, L. M., Rezende, W. C., Santos, P. E., Aguiar, R. et al. (2013). PID, LQR and LQR-PID on a quadcopter platform, *Informatics, Electronics & Vision (ICIEV), 2013 International Conference on*, IEEE, pp. 1–6.

Armstrong, R. D., Frome, E. L. und Kung, D. (1979). A revised simplex algorithm for the absolute deviation curve fitting problem, *Communications in Statistics-Simulation and Computation* **8**(2): 175–190. Taylor & Francis.

Baboolal, M. S. und Watson, G. A. (1981). Computational experience with an algorithm for discrete L_1 approximation, *Computing, Springer* **27**(3): 245–252.

Bancroft, J. (2010). *Multiple Inertial Measurement Unit Fusion for Pedestrian Navigation*, PhD thesis, Department of Geomatics Engineering, University of Calgary, Calgary, Alberta. UCGE Reports Number 20320.

Barrodale, I. (1968). L_1 approximation and the analysis of data, *Journal of the Royal Statistical Society: Series C (Applied Statistics)* **17**(1): 51–57.

Barrodale, I. und Roberts, F. D. (1973). An improved algorithm for discrete L_1 linear approximation, *SIAM Journal on Numerical Analysis* **10**(5): 839–848. SIAM.

Barrodale, I. und Roberts, F. D. (1974). Solution of an overdetermined system of equations in the L_1 norm [F4], *Communications of the ACM* **17**(6): 319–320. ACM.

Barrodale, I. und Roberts, F. D. (1980). Algorithm 552: Solution of the Constrained L_1 Linear Approximation Problem [F4], *ACM Transactions on Mathematical Software (TOMS)* **6**(2): 231–235. ACM.

Bartels, R. H. (1971). A stabilization of the simplex method, *Numerische Mathematik, Springer* **16**(5): 414–434.

Bartels, R. H. und Conn, A. R. (1980). Linearly Constrained Discrete L_1 Problems, *ACM Transactions on Mathematical Software (TOMS)* **6**(4): 594–608. ACM.

Bartels, R., Stoer, J. und Zenger, C. (1971). A realization of the simplex method based on triangular decompositions, *Linear algebra*, Springer, pp. 152–190.

Bebek, Ö., Suster, M. A., Rajgopal, S., Fu, M. J., Huang, X., Çavusoglu, M. C., Young, D. J., Mehregany, M., Van Den Bogert, A. J. und Mastrangelo, C. H. (2010). Personal navigation via high-resolution gait-corrected inertial measurement units, *IEEE Transactions on Instrumentation and Measurement* **59**(11): 3018–3027.

Becker, D. (2016). *Advanced calibration methods for strapdown airborne gravimetry*, PhD thesis, Technische Universität Darmstadt. Heft 51, Schriftenreihe der Fachrichtung Geodäsie. Fachbereich Bau- und Umweltingenieurwissenschaften. ISBN: 978-3-935631-40-2.

Becker, M. (2009). Status der Modernisierung von GPS und GLONASS und Perspektiven weiterer GNSS, *Zeitschrift für die Vermessungswesen* **134**(5): 297–305.

Benders, S., Wenz, A. und Johansen, T. A. (2018). Adaptive path planning for unmanned aircraft using in-flight wind velocity estimation, *2018 International Conference on Unmanned Aircraft Systems (ICUAS)*, IEEE, pp. 483–492.

Bittner, W. (2014). *Flugmechanik der Hubschrauber: Technologie, das flugdynamische System Hubschrauber, Flugstabilitäten, Steuerbarkeit*, Springer-Verlag. ISBN: 978-3-540-88972-4.

Blankenbach, J. und Norrdine, A. (2013). Indoor-Positionierung mit künstlichen Magnetfeldern, *zfv 138. Jg. 1/2013, Zeitschrift für Geodäsie, Geoinformation und Landmanagement* pp. 59–64.

Bloesch, M., Omari, S., Hutter, M. und Siegwart, R. (2015). Robust visual inertial odometry using a direct ekf-based approach, *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, IEEE, pp. 298–304.

Bloomfield, P. und Steiger, W. (1980). Least absolute deviations curve-fitting, *SIAM Journal on scientific and statistical computing* **1**(2): 290–301. SIAM.

Bodson, M. (2002). Evaluation of optimization methods for control allocation, *Journal of Guidance, Control, and Dynamics* **25**(4): 703–711.

Bodson, M. und Frost, S. A. (2011). Load balancing in control allocation, *Journal of Guidance, Control, and Dynamics* **34**(2): 380–387.

Bolić, M., Djurić, P. M. und Hong, S. (2004). Resampling algorithms for particle filters: A computational complexity perspective, *EURASIP Journal on Advances in Signal Processing* .

Borre, K., Akos, D. M., Bertelsen, N., Rinder, P. und Jensen, S. H. (2007). *A software-defined GPS and Galileo receiver: a single-frequency approach*, Springer Science & Business Media.

Bouabdallah, S. (2007). *Design and control of quadrotors with application to autonomous flying*, PhD thesis, École polytechnique fédérale de Lausanne.

Bouabdallah, S., Murrieri, P. und Siegwart, R. (2004). Design and control of an indoor micro quadrotor, *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, Vol. 5, IEEE, pp. 4393–4398.

Bouabdallah, S., Noth, A. und Siegwart, R. (2004). PID vs LQ control techniques applied to an indoor micro quadrotor, *Proc. of The IEEE International Conference on Intelligent Robots and Systems (IROS)*, IEEE, pp. 2451–2456.

Brescianini, D., Hehn, M. und D'Andrea, R. (2013). Nonlinear quadrocopter attitude control, *Technical report*, ETH Zürich. Department of Mechanical and Process Engineering.

Britting, K. (1971). *Inertial Navigation Systems Analysis*, Artech House Publishers. ISBN: 978-1608070-78-7.

Bry, A., Richter, C., Bachrach, A. und Roy, N. (2015). Aggressive flight of fixed-wing and quadrotor aircraft in dense indoor environments, *The International Journal of Robotics Research* **34**(7): 969–1002.

Burgard, W., Fox, D., Hennig, D. und Schmidt, T. (1996). Estimating the absolute position of a mobile robot using position probability grids, *Proceedings of the national conference on artificial intelligence*, pp. 896–901.

Carcanague, S. (2013). *Low-cost GPS/GLONASS Precise Positioning Algorithm in Constrained Environment*, PhD thesis, Université de Toulouse.

Caruso, M. J. (1997). Applications of magnetoresistive sensors in navigation systems, *Technical report*, Sensors and Actuators. SAE SP-1220.

Chang, X.-W., Yang, X. und Zhou, T. (2005). MLAMBDA: a modified LAMBDA method for integer least-squares estimation, *Journal of Geodesy* **79**(9): 552–565. Springer.

Chulliat, A., Macmillan, S., Alken, P., Beggan, C., Nair, M., Hamilton, B., Woods, A., Ridley, V., Maus, S. und Thomson, A. (2015). The US/UK World Magnetic Model for 2015-2020, *BGS and NOAA* .

Clasen, R. (1966). Techniques for automatic tolerance control in linear programming, *Communications of the ACM* **9**(11): 802–803.

Cramér, H. (1946). *Mathematical methods of statistics (PMS-9)*, Vol. 9, Princeton University Press.

Crassidis, J. L. (2006). Sigma-point Kalman filtering for integrated GPS and inertial navigation, *Aerospace and Electronic Systems, IEEE Transactions on* **42**(2): 750–756. IEEE.

Dantzig, G. B. (1990). Origins of the simplex method, *in* S. G. Nash (ed.), *A History of Scientific Computing*, ACM, New York, NY, USA, pp. 141–151. ISBN: 0-201-50814-1.

Dantzig, G. B. und Wolfe, P. (1960). Decomposition principle for linear programs, *Operations research* **8**(1): 101–111.

De Monte, P. S. (2015). Adaptive Trajektorienfolgeregelung für Quadrokopter basierend auf der L_1-adaptiven Regelungstheorie, *at-Automatisierungstechnik* **63**(8): 667–667.

Dellaert, F., Fox, D., Burgard, W. und Thrun, S. (1999). Monte carlo localization for mobile robots, *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*, Vol. 2, IEEE, pp. 1322–1328.

Delmerico, J. und Scaramuzza, D. (2018). A benchmark comparison of monocular visual-inertial odometry algorithms for flying robots, *Memory* **10**: 20.

Dennis, J. E. und Welsch, R. E. (1978). Techniques for nonlinear least squares and robust regression, *Communications in Statistics-Simulation and Computation* **7**(4): 345–359.

Devlin, S. J., Gnanadesikan, R. und Kettenring, J. R. (1981). Robust estimation of dispersion matrices and principal components, *Journal of the American Statistical Association* **76**(374): 354–362.

Domschke, W., Drexl, A., Klein, R. und Scholl, A. (2015). *Einführung in Operations Research*, Springer-Verlag.

Domschke, W., Drexl, A., Klein, R., Scholl, A. und Voß, S. (2014). *Übungen und Fallbeispiele zum Operations Research*, Springer-Verlag.

Dow, J. M., Neilan, R. E. und Rizos, C. (2009). The international GNSS service in a changing landscape of global navigation satellite systems, *Journal of geodesy* **83**(3-4): 191–198.

Dubins, L. (1961). On plane curves with curvature, *Pacific Journal of Mathematics* **11**(2): 471–481.

Ducard, G. und Hua, M.-D. (2011). Discussion and practical aspects on control allocation for a multirotor helicopter, *Conference on Unmanned Aerial Vehicle in Geomatics*, pp. 1–6.

Dutter, R. und Huber, P. J. (1981). Numerical methods for the nonlinear robust regression problem, *Journal of Statistical Computation and Simulation* **13**(2): 79–113.

Dydek, Z. T., Annaswamy, A. M. und Lavretsky, E. (2013). Adaptive control of quadrotor uavs: A design trade study with flight evaluations, *IEEE Transactions on control systems technology* **21**(4): 1400–1406.

Dziubek, N., Winner, H., Becker, M. und Leinen, S. (2012). Sensordatenfusion zur hochgenauen Ortung von Kraftfahrzeugen mit integrierter Genauigkeits-und Integritätsbewertung, *Tagungsband 5. Tagung Fahrerassistenz, München*.

Ebner, R. und Mark, J. (1978). Redundant integrated flight-control/navigation inertial sensor complex, *Journal of Guidance, Control, and Dynamics* **1**(2): 143–149.

Elkhatib, O. (2017). *Control allocation of a tilting rotor hexacopter*, B.S. thesis, ETH Zürich.

Faessler, M., Falanga, D. und Scaramuzza, D. (2017). Thrust mixing, saturation, and body-rate control for accurate aggressive quadrotor flight, *IEEE Robotics and Automation Letters* **2**(2): 476–482.

Faessler, M., Franchi, A. und Scaramuzza, D. (2018). Differential flatness of quadrotor dynamics subject to rotor drag for accurate tracking of high-speed trajectories, *IEEE Robot. Autom. Lett* **3**(2): 620–626.

Farebrother, R. (2013). *L_1-Norm and L∞-Norm Estimation: An Introduction to the Least Absolute Residuals, the Minimax Absolute Residual and Related Fitting Procedures*, Springer Science & Business Media.

Farrell, J. (2008). *Aided Navigation: GPS with High Rate Sensors*, McGraw-Hill, New York. ISBN: 978-0-071-49329-1.

Fisher, R. A. (1912). On an absolute criterion for fitting frequency curves., *Messenger of Mathmatics* **41**: 155–160.

Fletcher, R., Grant, J. und Hebden, M. (1971). The calculation of linear best Lp approximations, *The Computer Journal* **14**(3): 276–279.

Forsyth, D. A. und Ponce, J. (2011). *Computer Vision: A Modern Approach (2nd Edition)*, Prentice Hall.

Fox, D., Thrun, S., Burgard, W. und Dellaert, F. (2001). Particle filters for mobile robot localization, *Sequential Monte Carlo methods in practice*, Springer, pp. 401–428.

Foxlin, E. (2005). Pedestrian tracking with shoe-mounted inertial sensors, *IEEE Computer graphics and applications* **25**(6): 38–46.

Frangenberg, M., Stephan, J. und Fichter, W. (2015). Fast actuator fault detection and reconfiguration for multicopters, *AIAA Guidance, Navigation, and Control Conference*, p. 1766.

Frost, S. A. und Bodson, M. (2010). Resource balancing control allocation, *Proceedings of the 2010 American Control Conference* pp. 1326–1331.

Fuchs, H., Kedem, Z. M. und Naylor, B. F. (1980). On visible surface generation by a priori tree structures, *ACM Siggraph Computer Graphics*, Vol. 14, ACM, pp. 124–133.

Gelb, A. (1974). *Applied optimal estimation*, MIT press.

Goldberg, D. (1991). What every computer scientist should know about floating-point arithmetic, *ACM Computing Surveys (CSUR)* **23**(1): 5–48.

Gordon, N. J., Salmond, D. J. und Smith, A. F. (1993). Novel approach to nonlinear/non-Gaussian Bayesian state estimation, *IEE Proceedings F (Radar and Signal Processing)*, Vol. 140, IET, pp. 107–113.

Gray, R. A. und Maybeck, P. S. (1995). An integrated GPS/INS/baro and radar altimeter system for aircraft precision approach landings, *Aerospace and Electronics Conference, 1995. NAECON 1995., Proceedings of the IEEE 1995 National*, Vol. 1, IEEE, pp. 161–168.

Grewal, M. und Andrews, A. (2001). *Kalman filtering: theory and practice using MATLAB*, 3rd edn, John Wiley, New York. ISBN: 978-0-470-17366-4.

Grewal, M. S., Weill, L. R. und Andrews, A. P. (2007). *Global Positioning Systems, Inertial Navigation, and Integration*, John Wiley and Sons, New York. ISBN: 978-0-470-09971-1.

Guennebaud, G. und Jacob, B. (2018). Eigen C++ template library for linear algebra, version 3, `http://eigen.tuxfamily.org`. Accessed: 15-Okt-2018.

Gupta, N. und Hauser, R. (2007). Kalman filtering with equality and inequality state constraints, *Numerical Analysis Group, Oxford University Computing Laboratory, Univ. Oxford, Oxford, U.K., Tech. Rep. 07/18* .

Gustafsson, F. (2010). Particle filter theory and practice with positioning applications, *IEEE Aerospace and Electronic Systems Magazine* **25**(7): 53–82.

Håkansson, M., Jensen, A. B., Horemuz, M. und Hedling, G. (2017). Review of code and phase biases in multi-GNSS positioning, *GPS Solutions* **21**(3): 849–860.

Hampel, F. R., Ronchetti, E. M., Rousseeuw, P. J. und Stahel, W. A. (1986). *Robust statistics: the approach based on influence functions*, Vol. 196, John Wiley & Sons.

Hänsler, E. (2013). *Statistische Signale: Grundlagen und Anwendungen*, Springer-Verlag.

Härkegård, O. und Glad, S. T. (2005). Resolving actuator redundancy—optimal control vs. control allocation, *Automatica* **41**(1): 137–144.

Harrison, A. und Newman, P. (2011). Ticsync: Knowing when things happened, *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, IEEE, pp. 356–363.

Hawkins, D. M. (1980). *Identification of outliers*, Chapman and Hall, London.

He, X. und Jianye, L. (2002). Analysis of lever arm effects in GPS/IMU integration system, *Trans. Nanjing Univ. Aeronaut. Astronaut.* **19**(1): 59–64.

Hirschmüller, H. (2003). *Stereo Vision Based Mapping and Immediate Virtual Walkthroughs*, PhD thesis, De Montfort University, Leicester.

Hofmann-Wellenhof, B., Legat, K. und Wieser, M. (2003). *Navigation: Principles of Positioning and Guidance*, Springer, Wien London. ISBN: 3211008284.

Hofmann-Wellenhof, B., Lichtenegger, H. und Wasle, E. (2008). *GNSS–global navigation satellite systems: GPS, GLONASS, Galileo, and more*, Springer Verlag, Wien London. ISBN: 3211730125.

Hong, S., Lee, M. H., Chun, H.-H., Kwon, S.-H. und Speyer, J. L. (2006). Experimental study on the estimation of lever arm in GPS/INS, *IEEE Transactions on vehicular technology* **55**(2): 431–448.

Hota, S. und Ghose, D. (2014). Optimal trajectory generation for convergence to a rectilinear path, *Journal of Intelligent & Robotic Systems* **75**(2): 223–242.

Huber, P. J. (1964). Robust estimation of a location parameter, *The annals of mathematical statistics* **35**(1): 73–101.

Huber, P. J. (1973). Robust regression: asymptotics, conjectures and Monte Carlo, *The annals of statistics* pp. 799–821.

IEEE (2006). IEEE Standard Specification Format Guide and Test Procedure for Single-Axis Laser Gyros, *IEEE Std 647-2006 (Revision of IEEE Std 647-1995)* pp. 1–83.

Jekeli, C. (2001). *Inertial Navigation Systems with Geodetic Applications*, Walter de Gruyter, Berlin New York. ISBN: 3110159031.

Jäger, R. (2018). Multisensornavigation auf Bayes'scher Grundlage – Stand, Anwendungen und Entwicklungen, *Schriftenreihe des Studiengangs Geodäsie und Geoinformatik / Karlsruher Institut für Technologie, Studiengang Geodäsie und Geoinformatik* pp. 123–130. ISBN: 978-3-7315-0777-2.

Jäger, R., Diekert, J., Hoscislawski, A. und Zwiener, J. (2012). SIMA – Raw Data Simulation Software for the Development and Validation of Algorithms for GNSS and MEMS based Multi-Sensor Navigation Platforms, *Proceedings FIG Working Week, Rome, Italy, May 6-10, 2012*, International Federation of Surveyors (FIG). ISBN: 97887-90907-98-3.

Jäger, R., Diekert, J., Hoscislawski, A. und Zwiener, J. (2013a). Algorithmen und Systementwicklungen zur Navigation und Objektgeoreferenzierung, *17. Internationale Geodätische Woche Obergurgl 2013: Beiträge zur Tagung vom 17. bis 22. Februar 2013*, Vermessung und Geoinformation der Universität Innsbruck, Wichmann Verlag, pp. 252–255. ISBN: 978-3-87907-526-3.

Jäger, R., Diekert, J., Hoscislawski, A. und Zwiener, J. (2013b). Development and validation of the raw data simulation software (SIMA) for multi-sensor navigation platforms with emphasis on the part of GNSS simulation, *Proceedings of InterExpo-GeoSiberia 2013*, Siberian State Academy of Geodesy (SSGA). ISBN: 978-5-87693-607-3.

Jäger, R., Müller, T., Saler, H. und Schwäble, R. (2005). *Klassische und robuste Ausgleichungsverfahren*, Wichmann, Heidelberg.

Jäger, R. und Zwiener, J. (2016). Flugdynamik, Multisensor-Navigation und Steuerung skalierbarer Out-/Indoor-Multicopter UAV, *UAV 2016 – Vermessung mit unbemannten Flugsystemen, Beiträge zum 148. DVW-Seminar am 18. und 19. Februar 2016 in Bonn, Germany*, Vol. 82, DVW Gesellschaft für Geodäsie, Geoinformation und Landmanagement e.V., pp. 53–81. ISBN: 978-3-95786-067-5.

Johansen, T. A. und Fossen, T. I. (2013). Control allocation—a survey, *Automatica* **49**(5): 1087–1103.

Julier, S. J., Uhlmann, J. K. und Durrant-Whyte, H. F. (1995). A new approach for filtering nonlinear systems, *American Control Conference, Proceedings of the 1995*, Vol. 3, IEEE, pp. 1628–1632.

Junhuan, P. (2005). The asymptotic variance–covariance matrix, Baarda test and the reliability of L_1-norm estimates, *Journal of Geodesy* **78**(11): 668–682.

Kaplan, E. D. und Hegarty, C. (eds) (2005). *Understanding GPS: Principles and Applications*, 2 edn, Artech House.

Karbowski, A. (2015). Decomposition and parallelization of linear programming algorithms, *Progress in Automation, Robotics and Measuring Techniques*, Springer, pp. 113–126.

Kiesel, S. (2012). *GPS-Navigationssystem mit inertial gestütztem, vektoriellen Signal-Tracking*, PhD thesis, Karlsruher Institut für Technologie (KIT).

Kis, L. und Lantos, B. (2014). Development of state estimation system with INS, magnetometer and carrier phase GPS for vehicle navigation, *Gyroscopy and Navigation* **5**(3): 153–161. Springer.

Kálmán, R. E. (1960). A new approach to linear filtering and prediction problems, *Transactions of the ASME, Journal of Basic Engineering* **82**: 35–45.

Koenker, R. (1997). L_1 Computation: An Interior Monologue, *Lecture Notes-Monograph Series* pp. 15–32.

Kouba, J. und Héroux, P. (2001). Precise Point Positioning using IGS orbit and clock products, *GPS solutions* **5**(2): 12–28.

Kuipers, J. B. (1999). *Quaternions and Rotation Sequences: A Primer with Applications to Orbits, Aerospace and Virtual Reality*, Princeton University Press, Princeton, N.J. ISBN: 978-0-691-10298-6.

Leishman, G. J. (2006). *Principles of helicopter aerodynamics*, Cambridge University Press.

Leishman, J. G. (2000). *A history of helicopter flight*, University of Maryland. Accessed: 15-Okt-2018, `http://itlims-zsis.meil.pw.edu.pl/pomoce/WTLK/ENG/Sup/A_History_of_Helicopter_Flight.pdf`.

Lengyel, E. (2004). *Mathematics for 3D Game Programming and Computer Graphics*, 2nd edn, Charles River Media, Boston. ISBN: 1-58450-277-0.

Levy, L. J. (1997). The Kalman filter: navigation's integration workhorse, *GPS World* **8**(9): 65–71.

Leyffer, S. (2005). The return of the active set method, *Oberwolfach Reports* **2**(1): 107–109.

Lilliefors, H. W. (1967). On the Kolmogorov-Smirnov test for normality with mean and variance unknown, *Journal of the American Statistical Association* **62**(318): 399–402.

Ljung, L. (1999). *System identification: theory for the user*, Prentice-Hall information and system sciences series, 2nd edn, Prentice Hall, Upper Saddle River, NJ. ISBN: 978-0136566953.

Madyastha, V., Ravindra, V., Mallikarjunan, S. und Goyal, A. (2011). Extended Kalman filter vs. error state Kalman filter for aircraft attitude estimation, *AIAA Guidance, Navigation, and Control Conference*, p. 6615.

Maros, I. (2012). *Computational Techniques of the Simplex Method*, Vol. 61, Springer Science & Business Media.

Maybeck, P. S. (1979). *Stochastic models, estimation, and control*, Vol. 1 of *Mathematics in Science and Engineering*, Academic Press.

Maybeck, P. S. (1982). *Stochastic models, estimation, and control*, Vol. 2 of *Mathematics in Science and Engineering*, Academic Press.

McGee, L. A. und Schmidt, S. F. (1985). Discovery of the Kalman filter as a practical tool for aerospace and industry, *Technical Report 86847*, National Aeronautics and Space Administration (NASA).

Meister, O. (2010). *Entwurf und Realisierung einer Aufklärungsplattform auf Basis eines unbemannten Minihelikopters mit autonomen Flugfähigkeiten*, PhD thesis, Karlsruher Institut für Technologie (KIT). Logos Verlag, Berlin, ISBN: 978-3-8325-2603-0.

Merayo, J. M., Brauer, P., Primdahl, F., Petersen, J. R. und Nielsen, O. V. (2000). Scalar calibration of vector magnetometers, *Measurement science and technology* **11**(2): 120.

Metzger, J. (2006). *Optimierung des Akquisitions und Tracking-Verhaltens zentraler und modularer Terrainnavigationssysteme*, PhD thesis, Universität Fridericiana zu Karlsruhe.

Moler, C. B. (2004). *Numerical Computing with MATLAB*, Society for Industrial Mathematics, Philadelphia. ISBN: 0-898715-601.

Mueller, M. und D'Andrea, R. (2014). Stability and control of a quadrocopter despite the complete loss of one, two, or three propellers, *2014 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, pp. 45–52.

Müller, K. (1996). *Entwurf robuster Regelungen*, Teubner, Stuttgart.

Niemeier, W. (2008). *Ausgleichungsrechnung: Statistische Auswertemethoden*, Walter de Gruyter, Berlin. ISBN: 978-3-110190-557.

NOAA (1976). U.S. Standard Atmosphere, Washington, DC. National Oceanic and Atmospheric Administration (NOAA), National Aeronautics and Space Administration (NASA), United States Air Force (USAF).

Olson, E. (2011). AprilTag: A robust and flexible visual fiducial system, *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, pp. 3400–3407.

Oppenheimer, M. W., Doman, D. B. und Bolender, M. A. (2006). Control allocation for over-actuated systems, *Control and Automation, 2006. MED'06. 14th Mediterranean Conference on*, IEEE, pp. 1–6.

Paparazzi (2015). *Paparazzi Project: Mixing for arbitrary multirotor configurations*, École Nationale de l'Aviation Civile (ENAC). `http://paparazzi.enac.fr/wiki/RotorcraftMixing`, accessed: 15-Okt-2018.

Parkinson, B. W., Enge, P., Axelrad, P. und Spilker Jr, J. J. (1996). *Global Positioning System: Theory and applications, Volume II*, American Institute of Aeronautics and Astronautics.

Parviainen, J., Kantola, J. und Collin, J. (2008). Differential barometry in personal navigation, *Position, Location and Navigation Symposium, 2008 IEEE/ION*, IEEE, pp. 148–152.

Pejsa, A. J. (1974). Optimum skewed redundant inertial navigators, *AIAA Journal* **12**(7): 899–902.

Petovello, M. (2011). GNSS solutions: Clock offsets in GNSS receivers, *Inside GNSS* **6**(2): 23–25.

Petovello, M. G. (2003). *Real-time integration of a tactical-grade IMU and GPS for high-accuracy positioning and navigation*, PhD thesis, University of Calgary, Department of Geomatics Engineering.

Pittelkau, M. E. (2005). Calibration and attitude determination with redundant inertial measurement units, *Journal of Guidance, Control, and Dynamics* **28**(4): 743–752.

Plato, R. (2000). *Numerische Mathematik kompakt*, Vieweg+Teubner Verlag. ISBN: 978-3834802774.

Press, W. H., Flannery, B. P., Teukolsky, S. A. und Vetterling, W. T. (2007). *Numerical recipes 3rd edition: The art of scientific computing*, Cambridge University Press. ISBN: 978-0521880688.

Pretto, A. und Grisetti, G. (2014). Calibration and performance evaluation of low-cost IMUs, *Proc. of: 20th IMEKO TC4 International Symposium*, pp. 429–434.

Puls, T. (2011). *Lokalisations-und Regelungsverfahren für einen 4-Rotor-Helikopter*, PhD thesis, Universität Oldenburg.

Quan, Q. (2017). *Introduction to multicopter design and control*, Springer. ISBN: 978-981-10-3382-7.

Rabinowitz, P. (1968). Applications of linear programming to numerical analysis, *SIAM Review* **10**(2): 121–159.

Rauch, H. E., Striebel, C. und Tung, F. (1965). Maximum likelihood estimates of linear dynamic systems, *AIAA journal* **3**(8): 1445–1450.

Realini, E. (2009). *goGPS - free and constrained relative kinematic positioning with low cost receivers*, PhD thesis, Politecnico di Milano, Laboratorio di Geomatica.

Remondi, B. W. (2004). Computing Satellite Velocity using the Broadcast Ephemeris, *GPS Solutions* **8**: 181–183. Springer Berlin / Heidelberg.

Renaudin, V., Afzal, M. H. und Lachapelle, G. (2010). Complete triaxis magnetometer calibration in the magnetic domain, *Journal of sensors, Hindawi* **2010**. DOI: 10.1155/2010/967245.

Rossi, L. (2003). The LHC superconducting magnets, *Proceedings of the Particle Accelerator Conference, 2003. PAC 2003*, Vol. 1, IEEE, pp. 141–145.

Roth, J., Kaschwich, C. und Trommer, G. (2012). Improving GNSS attitude determination using inertial and magnetic field sensors, *Inside GNSS* **7**(1): 54–62.

Runge, C. (1895). Über die numerische Auflösung von Differentialgleichungen, *Mathematische Annalen* **46**(2): 167–178.

Rupalla, A. (2018). *Entwicklung eines Sensorkonzepts zur Umgebungserkennung für ein autonomes Lufttaxi*, Master Thesis, Universität Stuttgart.

Ryll, M., Bülthoff, H. H. und Giordano, P. R. (2013). First flight tests for a quadrotor UAV with tilting propellers, *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, IEEE, pp. 295–302.

Savage, P. G. (1998a). Strapdown inertial navigation integration algorithm design part 1: Attitude algorithms, *Journal of guidance, control, and dynamics* **21**(1): 19–28.

Savage, P. G. (1998b). Strapdown inertial navigation integration algorithm design part 2: Velocity and position algorithms, *Journal of Guidance, Control, and Dynamics* **21**(2): 208–221.

Savitzky, A. und Golay, M. J. (1964). Smoothing and differentiation of data by simplified least squares procedures, *Analytical chemistry* **36**(8): 1627–1639.

Savla, K., Frazzoli, E. und Bullo, F. (2005). On the point-to-point and traveling salesperson problems for dubins' vehicle, *Proceedings of the 2005, American Control Conference, 2005.*, IEEE, pp. 786–791.

Schilling, R. L. und Partzsch, L. (2014). *Brownian motion: an introduction to stochastic processes*, Walter de Gruyter. ISBN: 978-3110278897.

Schüler, T. (1997). *Untersuchungen zur GPS/INS-Integration für präzise Echtzeitanwendungen*, Diplomarbeit, Universität Hannover, Institut für Erdmessung.

Schmidbauer, H. (2013). *Abwickelbare Flächen: eine Konstruktionslehre für Praktiker*, Springer-Verlag.

Schönemann, E. (2013). *Analysis of GNSS raw observations in PPP solutions*, PhD thesis, Technische Universität Darmstadt. Heft 42, Schriftenreihe der Fachrichtung Geodäsie. Fachbereich Bau- und Umweltingenieurwissenschaften. ISBN: ISBN: 978-3-935631-31-0.

Schumacher, C. J. und Kumar, R. (2000). Adaptive control of UAVs in close-coupled formation flight, *American Control Conference, 2000. Proceedings of the 2000*, Vol. 2, IEEE, pp. 849–853.

Schumacker, R. A., Brand, B., Gilliland, M. G. und Sharp, W. H. (1969). Study for applying computer-generated images to visual simulation, *Technical report*, General Electric Co. Daytona Beach Fl. Apollo and Ground Systems.

Seeber, G. (1988). *Satellitengeodäsie: Grundlagen, Methoden und Anwendungen*, Walter de Gruyter, Berlin. ISBN: 3110100827.

Seleckỳ, M., Váňa, P., Rollo, M. und Meiser, T. (2013). Wind corrections in flight path planning, *International Journal of Advanced Robotic Systems* **10**(5): 248.

Serrano, L., Kim, D., Langley, R., Itani, K. und Ueno, M. (2004). A GPS velocity sensor: How accurate can it be? – a first look, *ION NTM*, pp. 875–885.

Shappell, S., Detwiler, C., Holcomb, K., Hackworth, C., Boquet, A. und Wiegmann, D. A. (2017). Human error and commercial aviation accidents: an analysis using the human factors analysis and classification system, *Human Error in Aviation*, Routledge, pp. 73–88.

Simon, D. und Chia, T. L. (2002). Kalman filtering with state equality constraints, *IEEE transactions on Aerospace and Electronic Systems* **38**(1): 128–136.

Späth, H. (1987). *Mathematische Software zur linearen Regression*, Oldenbourg, München. ISBN: 3-486-20375-4.

Stachniss, C. und Burgard, W. (2014). Particle filters for robot navigation, *Foundations and Trends® in Robotics* **3**(4): 211–282. ISBN: 978-1-60198-759-4.

Stengel, R. F. (1973). Some effects of bias errors in redundant flight control systems, *Journal of Aircraft* **10**(3): 150–156.

Stephan, J. und Fichter, W. (2017). Fast exact redistributed pseudoinverse method for linear actuation systems, *IEEE Transactions on Control Systems Technology* **27**(1): 451–458. DOI: 10.1109/TCST.2017.2765622.

Storøy, S. (1967). Error control in the simplex-technique, *BIT Numerical Mathematics* **7**(3): 216–225.

Su, J. und Cai, K.-Y. (2011). Globally stabilizing proportional-integral-derivative control laws for rigid-body attitude tracking, *Journal of Guidance, Control, and Dynamics* **34**(4): 1260–1264.

Swope, W. C., Andersen, H. C., Berens, P. H. und Wilson, K. R. (1982). A computer simulation method for the calculation of equilibrium constants for the formation of physical clusters of molecules: Application to small water clusters, *The Journal of Chemical Physics* **76**(1): 637–649.

Takasu, T. (2012). *Kalman Correction Step Equations*, Tomoji Takasu Online Memorandum, `http://gpspp.sakura.ne.jp/diary201207.htm`. Accessed: 12-Okt-2018.

Takasu, T. und Yasuda, A. (2009). Development of the low-cost RTK-GPS receiver with an open source program package RTKLIB, *Proceedings of international symposium on GPS/GNSS, Jeju, Korea*.

Tanigawa, M., Luinge, H., Schipper, L. und Slycke, P. (2008). Drift-free dynamic height sensor using MEMS IMU aided by MEMS pressure sensor, *Positioning, Navigation and Communication, 2008. WPNC.*, IEEE, pp. 191–196.

Tebboth, J. R. (2001). *A computational study of Dantzig-Wolfe decomposition*, PhD thesis, University of Buckingham.

Tedaldi, D., Pretto, A. und Menegatti, E. (2014). A robust and easy to implement method for IMU calibration without external equipments, *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, IEEE, pp. 3042–3049.

Teller, S. J. (1992). *Visibility computations in densely occluded polyhedral environments*, PhD thesis, University of California at Berkeley.

Teunissen, P. (2004). Integer least-squares, *In Proc. V Hotine-Marussi Symposium on Mathematical Geodesy*, Vol. 127 of *International Association of Geodesy Symposium*, Springer, Matera, Italy, pp. 69–80.

Teunissen, P. J. (1995). The least-squares ambiguity decorrelation adjustment: a method for fast GPS integer ambiguity estimation, *Journal of Geodesy, Springer* **70**(1-2): 65–82.

Thrun, S. (1998). Bayesian landmark learning for mobile robot localization, *Machine learning, Springer* **33**(1): 41–76.

Thrun, S. (2002). Particle filters in robotics, *Proceedings of the Eighteenth conference on Uncertainty in artificial intelligence*, Morgan Kaufmann Publishers Inc., pp. 511–518.

Thrun, S., Burgard, W. und Fox, D. (2005). *Probabilistic robotics*, MIT press.

Titterton, D. und Weston, J. (2004). *Strapdown Inertial Navigation Technology*, Vol. 17, Peter Peregrinus Ltd.

Trippel, T., Weisse, O., Xu, W., Honeyman, P. und Fu, K. (2017). WALNUT: Waging doubt on the integrity of MEMS accelerometers with acoustic injection attacks, *Security and Privacy (EuroS&P), 2017 IEEE European Symposium on*, IEEE, pp. 3–18.

Tu, Y., Lin, Z., Lee, I. und Hei, X. (2018). Injected and delivered: fabricating implicit control over actuation systems by spoofing inertial sensors, *27th USENIX Security Symposium (Security 18)*, pp. 1545–1562.

Unbehauen, H. (2011). *Regelungstechnik*, Vol. 3: Identifikation, Adaption, Optimierung, Vieweg + Teubner, Wiesbaden. ISBN: 978-3-8348-1419-7.

Van Der Merwe, R. (2004). *Sigma-point Kalman filters for probabilistic inference in dynamic state-space models*, PhD thesis, Oregon Health & Science University.

Van Der Merwe, R. und Wan, E. (2004). Sigma-point Kalman filters for integrated navigation, *Proceedings of the 60th Annual Meeting of the Institute of Navigation (ION)*, pp. 641–654.

VectorNav (2016). Inertial Measurement Units and Inertial Navigation, *VectorNav web page*. Accessed: 15-Okt-2018.
URL: *http://www.vectornav.com/support/library?id=76*

Verlet, L. (1967). Computer experiments on classical fluids. I. Thermodynamical properties of Lennard-Jones molecules, *Physical review, APS* **159**(1): 98.

Waegli, A., Guerrier, S. und Skaloud, J. (2008). Redundant MEMS-IMU integrated with GPS for performance assessment in sports, *Proceedings of IEEE/ION PLANS 2008*, IEEE, pp. 1260–1268.

Wagner, H. M. (1959). Linear programming techniques for regression analysis, *Journal of the American Statistical Association* **54**(285): 206–212.

Wagner, J. F. (2003). Zur Verallgemeinerung integrierter Navigationssysteme auf räumlich verteilte Sensoren und flexible Fahrzeugstrukturen, *Fortschrittberichte VDI : Reihe 8, Meß-, Steuerungs- und Regelungstechnik; 1008*, VDI-Verlag, Düsseldorf. ISBN: 3-18-500808-1.

Wang, L.-S., Chiang, Y.-T. und Chang, F.-R. (2002). Filtering method for nonlinear systems with constraints, *IEEE Proceedings-Control Theory and Applications* **149**(6): 525–531.

Weiss, S. M. (2012). *Vision based navigation for micro helicopters*, PhD thesis, ETH Zürich.

Wendel, J. (2011). *Integrierte Navigationssysteme: Sensordatenfusion, GPS und inertiale Navigation*, Oldenbourg, München. ISBN: 978-3-486-70439-6.

Wendel, J., Maier, A., Metzger, J. und Trommer, G. (2005). Comparison of extended and sigma-point Kalman filters for tightly coupled GPS/INS integration, *AIAA Guidance, Navigation, and Control Conference and Exhibit*, AIAA, San Francisco, California. DOI: 10.2514/6.2005-6055.

Wendel, J., Meister, O., Schlaile, C. und Trommer, G. F. (2006). An integrated GPS/MEMS-IMU navigation system for an autonomous helicopter, *Aerospace Science and Technology* **10**(6): 527–533.

Wicki, F. (1992). *Robuste M-Schätzer und Zuverlässigkeit*, Bericht 190, Inst. für Geodäsie und Photogrammetrie, ETH Zürich.

Wicki, F. (1998). *Robuste Schätzverfahren für die Parameterschätzung in geodätischen Netzen*, PhD thesis, ETH Zürich.

Williams, H. P. (2009). Integer programming, *Logic and Integer Programming*, Springer, pp. 25–70.

Winston, W. L. und Goldberg, J. B. (2004). *Operations Research: Applications and Algorithms*, Vol. 3, Duxbury press Boston.

Wolfe, J. M. (1979). On the convergence of an algorithm for discrete L_p approximation, *Numerische Mathematik* **32**(4): 439–459.

Woodbury, M. A. (1950). Inverting modified matrices, *Memorandum report* **42**: 106. Statistical Research Group, Princeton University.

Woodman, O. J. (2007). An introduction to inertial navigation, *University of Cambridge, Computer Laboratory, Tech. Rep. UCAMCL-TR-696* **14**: 15.

Wu, J.-T., Wu, S. C., Hajj, G. A., Bertiger, W. I. und Lichten, S. M. (1992). Effects of antenna orientation on GPS carrier phase, *Astrodynamics 1991*, pp. 1647–1660.

Xu, G. und Xu, Y. (2016). *GPS: theory, algorithms and applications*, Springer.

Xu, P., Cannon, E. und Lachapelle, G. (2010). Mixed integer programming for the resolution of GPS carrier phase ambiguities, Technical Report Nr.2000.2, Department of Geodesy and Geoinformatics, Stuttgart University, arXiv preprint arXiv:1010.1052.

Yeh, Y. C. (1996). Triple-triple redundant 777 primary flight computer, *Proceedings Aerospace Applications Conference*, Vol. 1, IEEE, pp. 293–307.

Zarchan, P., Parkinson, B. W. et al. (1996). *Global Positioning System: Theory and Applications, Volume I*, American Institute of Aeronautics and Astronautics.

Zeimetz, P., Becker, M., Kuhlmann, H., Schön, S. und Wanninger, L. (2011). Berücksichtigung von Antennenkorrekturen bei GNSS-Anwendungen, *DVW Merkblatt 1-2011* **1**: 1–10.

Zhang, L. und Schwieger, V. (2018). Investigation of a L1-optimized choke ring ground plane for a low-cost GPS receiver-system, *Journal of Applied Geodesy* **12**(1): 55–64.

Zhao, Y. (2016). Performance evaluation of cubature Kalman filter in a GPS/IMU tightly-coupled navigation system, *Signal Processing* **119**: 67–79.

Zhou, M. und Prasad, J. (2014). 3d minimum fuel route planning and path generation for a fuel cell powered uav, *Unmanned Systems* **2**(01): 53–72.

Zwiener, J. (2012). *Entwurf und Implementierung eines Algorithmus zur Quaternionen-basierten GNSS/-MEMS Navigationszustandsschätzung mit tiefer GNSS Kopplung*, Master Thesis, Fakultät Geomatik, Hochschule Karlsruhe – Technik und Wirtschaft.

Zwiener, J. und Diekert, J. (2012). Multisensorfusion zur autonomen Navigation und Objektgeoreferenzierung, *Geomatik aktuell – Präzise Navigation und mobile Geodatenerfassung – Out- und Indoor*, Vol. 7, Reihe B, Hochschule Karlsruhe – Technik und Wirtschaft, pp. 33–41. ISBN: 978-3-89063-106-6.

Zwiener, J., Lorenz, A. und Jäger, R. (2014). Seamless Indoor/Outdoor Navigation with different Custom Low-Cost Foot-Mounted MEMS/GNSS Sensors, *Proceedings of Interexpo Geo-Siberia 2014*, Siberian State Academy of Geodesy (SSGA), pp. 83–97. ISBN: 978-5-87693-715-5.

Zwiener, J., Lorenz, A. und Jäger, R. (2015). Flight Control and Navigation for Scalable and Arbitrarily Dimensioned UAV and Manned Multicopters, *Proceedings of Interexpo GeoSiberia April 2015*, Siberian State Academy of Geodesy (SSGA), pp. 98–108. ISBN: 978-5-87693-803-9.

Curriculum Vitæ

Persönliche Daten

Name	Jan Zwiener
Geburtstag	12.05.1982
Geburtsort	Sindelfingen, Deutschland
Familienstand	verheiratet (09/2015), zwei Kinder (Zwillinge, 06/2016)
Nationalität	Deutsch

Lebenslauf

1992 - 1998	*Mittlere Reife* an der Realschule Calw.
1998 - 2000	*Fachhochschulreife* am Berufskolleg der Gottlieb-Daimler-Schule in Sindelfingen. Abschluss mit Auszeichnung.
2001 - 2004	Ausbildung zum *staatlich geprüften Informatiker* an der Akademie für Datenverarbeitung in Böblingen u. Robert Bosch GmbH am Standort Leonberg.
2004 - 2007	*System Engineer* bei der teXXmo GmbH, Böblingen.
2007 - 2011	Studium *Vermessung und Geomatik* an der Hochschule Karlsruhe (HSKA).
2009	Praxissemester bei der Robert Bosch GmbH: *Verbesserung der GPS Genauigkeit. für autonome Rasenmäher durch Precise Point Positioning (PPP)*.
2010 - 2011	Abschlussarbeit (B. Sc.) am Europäischen Kernforschungszentrum CERN. Titel der Arbeit: *3-D Integrated Network Adjustment with a Parametric Height Reference Surface*. Preis für die Thesis erhalten von Arbeitskreis Beratende Ingenieure e.V./Bund Deutscher Baumeister, Architekten und Ingenieure.V. (abv/BDB).
2011 - 2012	Master Studium (M. Sc.) *Geomatik* an der Hochschule Karlsruhe (HSKA). Preis für herausragende Studienleistungen im Master Studiengang Geomatik.
2014	Gewinner Landespreis mit der Forschungsgruppe Navka bei der European Satellite Navigation Competition (ESNC) 2014.
2012 - 2016	*Wissenschaftlicher Mitarbeiter* am Institut für Angewandte Forschung (IAF) und Doktorand an der Technischen Universität Darmstadt.
2016 - 2017	*Senior Software Engineer* bei der e-volo GmbH.
seit 2017	*Head of Flight Control and Navigation* bei der Volocopter GmbH.

Lehre

2013 - 2014	Praktikumsbetreuung *Geodatenerfassung 2* an der Hochschule Karlsruhe (HSKA) im Studiengang Geoinformationsmanagement (Bachelor).
seit 2015	Vorlesung *Mathematische Modelle der Sensorfusion* an der Hochschule Karlsruhe (HSKA) im Studiengang Geodäsie und Navigation (Bachelor).

Index